TO STUDY AND ANALYSIS OF OPTIMIZATION TECHNIQUES AND THEIR APPLICATION TO EVOLVE SOME ALGORITHM TO PERFORMANCE ENHANCEMENT OF COMMUNICATION SYSTEM

A

THESIS

SUBMITTED IN FULFILLMENT

OF THE AWARD OF THE DEGREE OF

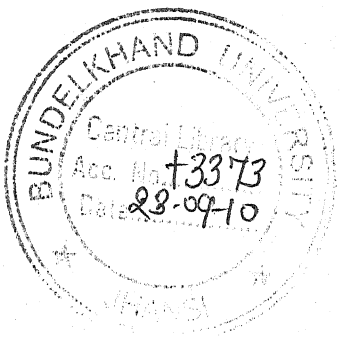
DOCTOR OF PHILOSOPHY

IN

MATHEMATICS

By:

MUDIT BANSAL



Under the supervision of:

DR. AVANISH KUMAR
Senior Lecturer
Department of Mathematical Sciences
and Computer Applications

BUNDELKHAND UNIVERSITY
JHANSI-284128, UP (INDIA)

November, 2007

Dedicated to My Mother

Late Smt. JYOTI BANSAL

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **"To Study and Analysis of Optimization Techniques and their Application to Evolve some Algorithm to Performance Enhancement of Communication System"** in fulfillment of the requirement for the award of the Degree of **Doctor of Philosophy** in **Mathematics**, submitted in the Department of Mathematical Sciences and Computer Applications, Bundelkhand University, Jhansi is an authentic record of my research work, under the kind supervision of **Dr. Avanish Kumar**, Senior Lecturer, Department of Mathematical Sciences and Computer Applications, Bundelkhand University, Jhansi.

It is also certify that I have not submitted the matter embodied in this thesis for the award of any other degree.

Mudit

(Mudit Bansal)

Dated: 27th Nov 2007

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge. It is also certify that he has put a minimum number of attendance i. e. 200 days in the department as required by the university.

Supervisor

Kumar 27/XI/07

(Dr. Avanish Kumar)
Senior Lecturer
Department of Mathematical Sciences
and Computer Applications
Bundelkhand University
Jhansi-284128, UP(INDIA)

ACKNOWLEDGEMENT

The present research work entitled **"To Study and Analysis of Optimization Techniques and their Application to Evolve some Algorithm to Performance Enhancement of Communication System"** in fulfillment of the requirement for the award of the Degree of Doctor of Philosophy in Mathematics, is undertaken the very kind supervision of **Dr Avanish Kumar**, Senior Lecturer, Department of Mathematical, Sciences and Computer Applications, Bundelkhand University, Jhansi. His encouragement, inspirational guidance invaluable suggestions and continued support have enabled me to complete this piece of research.

I wish to express my sincere thank to **Prof. V. K. Sehgal**, Dean of Science, Faculty of Science, Director, Institute of Computer Sciences and Information Technology and Head, Department of Mathematical Sciences and Computer Applications, Bundelkhand University, Jhansi for providing me all the necessary facilities available.

My heartiest thanks are due to **Dr. Alok Kumar Verma**, all Faculty Members and Staff Members, Department of Mathematical Sciences and Computer Applications, Bundelkhand University, Jhansi for providing all necessary help at their end.

My heartiest thanks are also due to **Dr. P. M. Bhandari**, Director, **Prof. Subhash C. Jain**, Head, **Dr. Lokesh Bajpai, Prof. Sandeep Jain, Prof. Sanjay Jain, Prof. Sanjay Katare, Mr. A. K. Yadav, Mr. P. K. Ahirwar** and all my colleagues of the Department of Mechanical Engineering, and **Prof. Pramod Sharma**, Department of Applied Mechanics, Samart Ashok Technological Institute (Engineering College), **Prof. Amit Jain,** Department of Electronics &

Telecommunication Samart Ashok Technological Institute (Polytechnic)Vidisha, MP for providing me all the necessary help required time to time.

I wish to express my thank to **Dr. D. P. Goyal,** Professor & Head **Dr Rajendra Dubey**, and all faculty members of Department of Applied Mathematics and Computer Science, Samart Ashok Technological Institute (Engineering College), Vidisha, MP for their continued support in during this study.

I am very thankful to **Prof. M. P. Singh,** Head, Department of Mathematics, Gurukula Kangri Vishwavidyalaya, Hardwar, and **Prof. Vinod Kumar,** Department of Computer Science, Gurukula Kangri Vishwavidyalaya, Hardwar, for their guidance and motivation.

I owe my sincere respect to **Prof. P. N. Pandey**, Allahabad, **Prof. G. C. Sharma,** Agra, **Prof. R. C. Mittal** and **Prof. Rama Bhargava**, Roorkee for their motivation and inspiration.

My heartiest thanks are due to my dearest senior **Dr. P. K. Yadav**, Senior Scientist, CBRI, Roorkee, **Dr. Manisha Sharma**, Punjab University, Chandigarh, **Ms. Monika Saroha**, DIT, Dehradun, **Mr. Harendra Kumar,** Roorkee, **Mr. Kuldeep Sharma,** KIET, Gajiyabad, **Mr. Pankaj Agrawal, Mr. Rajeev Jain, Mr. Chitrang Jain, Mr. Bhupendra Singh, Mr. Shekhar Bhargava, Mr. Ashish Jain, Dr. Rahul Jain(MD), Mr. Anand Nigudkar, Mr. Brij Mohan Solanki, Mr. Neeraj Maheshwari,** Vidisha and **all friends** and **well wishers**, who helped me to the maximum extent of their capabilities. Special thank to **Prof. A. K. Ronghe**, S. S. L. Jain (PG) College, and **Prof. Vijay Shrivastava**, Vidisha, MP who has given me the external motivation and support in fulfilling this task.

I was highly motivated and benefited by the books, research papers, Magazines, Research Journals and other related material which were easily made

available by the librarians and other staff members of various libraries such as, GKV, Hardwar, IIT, Roorkee, IIT, Kanpur, INSDOC, New Delhi, CBRI, Roorkee, BIET, Jhansi, Bundelkhand University, Jhansi, SATI, Vidisha, BHU, Varansi, Allahabad University, Allahabad. I owe my thanks to the librarians and other staff members of these libraries.

I would also like to express my appreciation to those Indian and Foreign Research workers who made available some of their research papers which proved to be very help full in the present work. My deep appreciation to those organizers those organize various Conferences and Seminars, and given me the opportunity to present my research work.

I must not fail to record my appreciation to **Mr. Bhupendra Bansal**, father, **Mrs. Santosh Bansal**, mother, **Mrs. Gulab Agarwal**, mother-in-law, **Mrs. Rachna Bansal**, wife, **Mr. Vishal Bansal**, **Mr. Amit Bansal**, brothers, **Mrs. Babita Mittal**, **Mrs. Monika Mittal**, **Mrs. Sweta Bansal**, sisters, **Mrs. Aarti Yadav**, Sister-in-law, **Mrs. Suman Bansal**, bhabhi, and all the in-laws who encouraged me all the time and did not loose her patience in spite of lot of inconvenience caused to them by me during the completion of this work. Thanks to lovable my daughter **Bhumi** and niece **Manya**. I sincerely thanks to God.

Mudit

(Mudit Bansal)

Dated: 27th NOV 2007

Place: JHANSI

CONTENTS

CHAPTER-1

INTRODUCTION

1.1. INTRODUCTION TO OPTIMIZATION TECHNIQUES

1.1.1. An Overview

During the last 1930s in England and early 1940s in United States, the use of scientific methods was extensively made to analyze optimization problems. World War II challenged both countries at this time to develop optimal solution for allocation, transportation and multi-variable type problems. The modeling techniques studied by operations researchers in 1940s and 1950s usually required algebra or calculus for solution purposes. Mathematical Programming is one of the most powerful tools for optimization of communication parameters and it is still used today by the researchers, to describe the structuring of mathematical symbols into a model or program. These problems first arose in the field of economics where allocation problems had been a subject of deep interest.

During World War II, a group of researchers sought to solve allocation type problems for the United States Air Force. One of the members of this group formulated and devised a solution procedure in 1947 for linear programming type problems. This solution procedure, called the simplex method, marked the beginning of the field of study called Mathematical Programming. It is useful for finding an optimal solution to a

complex problem involving many interrelated variables. It also simplifies exposition of many network problems and helps in finding optimal results which management must implement to realize its objectives. In 1972, [TURB 1972], surveyed a large number of United States Corporations on their use of operation research activities. One of his conclusions was that individuals within organization were more often viewing decisions making situations as a management science type problem requiring some types of mathematical modelling, than ever before observed in the past.

Markland and Newett [MARK 1972] voiced concern in their study on the misuse of mathematical programming and future problems that can be caused by the ineffective use of management science practitioners who fail to consider implementation as a part of every study undertaken in management science. Some surveys [RADN 1973, GRAY 1973, FABO 1976] sought specific information on the use of mathematical programming techniques. One of these surveys conducted by Fabozzi and Valaente [FABO 1976] examined the use of mathematical programming methodology and where in the organization it was used.

A similar study was conducted by Ledbetter and Cox [LEDB 1977]. They found that many organizations were using the mathematical programming techniques as reported by Fabozzi and Valaente as well as other management science methodologies. An important area of development of mathematical programming concerns with the use of computer in operation research. More and more computer software are being developed every day making the computational aspects of mathematical programming less complex. Time-sharing and interactive computer systems are also influencing a de-

emphasis on computational experience and more on problem formulation. During the last decade, mathematical programming approaches have gained immense importance for solving task allocation and load balancing problems in communication systems [CHU 1980a, CHU 1980b, MA 1982, SINC 1987, and REID 1994].

1.1.2. Mathematical Programming Problems

During the last decade, mathematical programming techniques have been recognize as the most powerful methods for modelling and analyzing several kind of problems. Many research problems are formulated in the form of mathematical models, which describe the quantitative features of all types of problems. Mathematical programming techniques are used for the formulation and solution of research problems by systematic planning of various activities. The basic problem in mathematical programming is to find the unknown values of some variables, which will optimize the value of the objective function subject to a set of constraints. Most of the mathematical programming problems can be formulated in the following general form [KWAK 1987]:

Maximize (*or M*inimize) $Z = \sum_{j=1}^{n} c_j x_j$

subject to $\sum_{j=1}^{n} a_{ij} x_j <=> b_i$ (for i $=1,2,.....,$ m)

and $x_j \geq 0$ ($forj = 1,2,.....,n$)

Where,

Z	=	value of the objective function which measures the effectiveness of the decision choice,
x_j	=	unknown variables that are subject to the control of the decision maker,
c_j	=	unit profit contribution of an output or cost of an input which is known,
a_{ij}	=	production (or technical) coefficients that are known, and
b_i	=	available resources in limited supply.

Production of goods, in any organization, requires productive resources, which are in short supply in the real world, and they are, therefore, restricted resources. These restricted resources are expressed as equalities or inequalities in mathematical programming models. The decision maker's goal is to find the values of the decision variables within limits and to optimize the value of the objective function.

1.2. INTRODUCTION TO DATA COMMUNICATIONS

1.2.1. The Need of Study

The reasons for studying data communications can be summed up in the occupational history of the United States. In the 1800s they were an agricultural society dominated by farmers. By the 1900s they had moved into an industrial society dominated by labor and management. Now, as they approach the twenty-first century, they clearly have moved into the information society, which do computers, data communications, and highly skilled individuals who use brainpower instead of physical power dominate. The

industrial society has reached its zenith, and the communication/computer era, started in mid 1950s, which is dubbed the information society, is advancing rapidly.

In an information society dominated by computers and communications, value is increased by knowledge as well as by the speed of movement of that knowledge. This new information economy will completely destroy Ricardo's labor theory of value, because, in such a society, what increases value is not the labor of individuals, but information.

The main stream of the information age is a communication network. The value of a high-speed data communication network that transmits knowledge/information is that it brings the message sender and the message receiver closer together in time. For example, in the 1800s it might have taken several weeks for specific information to reach the United States from England. By the 1900s it could be transmitted within an hour. Today, with modern data communication systems, it can be transmitted within seconds.

Finally, the transition from an industrial to an information society means that we have to learn many new technologically based skills. The study of data communications has become a basic tool that can be used throughout our lifetime. We incorporate our knowledge of data communications into several careers such as circuit designer, programmer, business system application developer, communication specialist, and business manager.

1.2.2. Evolution

Today we take data communications for granted, but it was early pioneers like Samuel Morse, Alexander Graham Bell, and Thomas Edison who developed the basic electrical and electronic systems that ultimately became today's voice and data communication networks.

When the telephone arrived, it became the accepted communication device that everyone wanted. Several technological enhancements were made in telephonic technology from 1837 to 1951 and it was in 1951 that the first direct long distance customer dialing began. The first international satellite telephone call was sent over the Telstar satellite in 1962. In 1963, touch-tone telephones began to be marketed. Their push buttons were easier to use than rotary dials, and they became quite popular. By 1965, there was widespread introduction of commercial international telephone service by satellite.

Picture phone service, which allows users to see as well as talk with one another, began operating in 1969. All through the 1970s there were many arguments and court cases regarding the monopolistic position that A T &T held over other companies that wanted to offer communication services. The litigation led to the litigation led to the divestiture of AT&T on January 1, 1984.

During 1983-84 the newer cellular telephone networks supplanted traditional radio telephone-type calls. Integrated Services Digital Networks [ISDN] began serving

the public in 1986. These networks allow the simultaneous transmission of voice, data, and video images [FITZ 1988].

During the 1987 there was considerable competition in both the voice and data communication markets as a number of independent companies began to sell communication services in a manner similar to that of automobile marketing. And now we have smaller and less expensive portable telephones to carry around everywhere.

1.2.3. The Indian Scenario

India is a resource rich fast developing country where telecommunication has passed from the stage of convenience to essentiality. Computers have entered in a big way in Indian society. As on March 31 1986, it was estimated that the country had about 3050 super, mainframes and minis and 7000 micros and personal computers. The number of microcomputers and personal computers is growing fast. The idea of computer networking has been widely accepted in government organizations. The National Informatics Center [NIC] has been doing this for planning purposes to Govt. of India. NICNET computer and communication facilities have provided nation-wide links to make the computing resources available at the places from where the information emanates [AGAR 1995].

The Indian society has now clearly realized the importance of communications, which may be in its various forms including telephones, memorandums, telex, mail, reports etc. Attempts have been made by the organizations like Telecommunications

Research Center [TRC] to produce prototype model of several important communication systems. Electronic mail, facsimile equipment, modems, teletext voice mail etc. have been developed, integration of computer communication with special digital switches, ISDN, has also been tried. This has put India in the forefront of telecommunication services. The Govt. of India has realized that without a better and secures telecom system the fruits of development can never be fairly distributed, be it in education, economics, rural development or international trade and banking.

The development of efficient communication systems has opened new vistas in electronic industries. The OSI technology is coming up fast and fiber optics technology has also entered into the scene. Optical fiber computer communication networks [Fibre LANs] are now being developed in India. To meet the telematics challenge in the country, the IITs and some university departments have begun setting up facilities to train the scientists and the technicians, engaged in futuristic research and develop new systems in communication technology.

1.3. COMMUNICATION SYSTEM

1.3.1. Meaning of Communication System

The on-set of the microprocessor technology has made the Communication System [CS] economically viable and attractive for many applications of computer. However, many problem areas in communication system are still in their primitive development stages. CS is increasingly drawing attention, yet has a meaning that is not understood.

In a CS several computers interconnected in some fashion such that a program or procedure utilizes this distributed but combined power and gets executed in real time. The term has different meanings with regard to different systems, because processors can be interconnected in many ways for various reasons. In its most general form, the word distribution implies that the processors are fixed in geographically separated locations. Occasionally, the term is also applied to an operating environment using multiple mini-computers not connected with each other with the help of physical communication lines but are connected through satellite. A user-oriented definition [BHUT 1994, SITA 1995] of distributed computing is "Multiple Computers, utilized cooperatively to solve problems".

1.3.2. Main Aspects of Communication Systems

While addressing task allocation issues in a CS, the following are important aspects need to be consideration:

(i) **Multiplicity of resources:** There are a number of resources, in particular, processors. Homogeneity of physical resources is not essential. A system may have processors with identical characteristics and capabilities.

(ii) **Dispersion:** The resources in the system are physically or logically distributed. AU the processors are independent and tied together by communication links. The communication links provide means for transferring information between the processors.

(iii) **Control:** All the processors in the system are autonomous and there is no master and slave relation among the processors. For the user, the collection of processors should be invisible, the multiple processor system should appear as virtual uniprocessor, and users need not know on which machine their programs are running and where their files are stored.

(iv) **Transparency:** All the processors in the system are autonomous and there is no master-slave relationship among them. The fact there are several co-operating processors in the system must be invisible (transparent) to the user and a single image id presented to the user. The system should appear as a uniprocessor system and users need not know on which machine their programs are executing and where the data or files stored.

(v) **Flexibility:** Flexibility is an important aspect of a distributed system. It means that should be easy to incorporate changes in a user transparent manner or with minimum interruption to the user. Further, in every system, new functionalities are to be added from time to time to make it more powerful and easy to use. Therefore, It should be easy to add new services to the system.

(vi) **Performance:** The performance of the distributed system must be at least as good as a centralize system. That is when a particular application is run on distributed system, Its overall performance should be better than or at least equal to that of running the same application on a single processors system.

(vii) **Scalability:** Scalability refers to the capability of a system to adapt to increased service load. A distributed system must able to cope with the growth of nodes and users in the system.

(viii) **Security:** Security must be provided in the system to prevent destruction and unauthorized access so that user can trust on the system and rely on it.

1.3.3. Machine Size Vs Instruction Execution

Communication Systems provides much faster execution by facilitating parallel execution of tasks. A major driving force towards distributed processing is the cost of small processors. Until the spread of mini computers in the early 1970s, a commonly accepted rule was Grosch's Law, which said, *"The cost per machine instruction executed is inversely proportional to the square of the size of the machine "*. Grosch's Law became questionable in the 1970s. Even some people suggested that it had been reversed because the cost per instruction on some mini-computers was lower than on large computers and on microprocessors was lower than mini-computers. The reason is related to the use of Very Large Scale Integrated [VLSI] circuits, which can be mass-produced economically. Their development cycle is much shorter than that of large machines.

1.3.4. The Logical Vs Physical Design

The implementation of communication systems depends both on logical and physical premises. The logical functions center on the procedural design for channeling the flow of data and controlling the physical facility throughput of the projected system

configuration. The object' of the physical functions is the design of the hardware and hard-software [firmware] devices to provide a specific level of capability. Functions will be distributed both to machines in the computer center and too many machines at user locations. This trend of the distribution of processing is continuing because software usage of machine instructions per second is growing much faster than the development of higher speed machines. The CS is motivated by the need for cost reduction in tasks executing.

1.3.5. Distributed Vs Parallel Computing

In some computer networks, the control mechanism is mostly centralized. In others, they are mostly distributed. Where purely centralized control exists, loss of the center puts the entire network out of action. With the distributed control any portion of network can be destroyed and the rest will continue to function. Although computation speed has increased several folds over the past three decades of computing, the user demand for faster speeds is growing at a much faster rate. One of the approaches to meet the growing demand of faster computation is to use parallel processing. Parallel computers, which emphasize parallel processing, may be employed. Parallel computers are available with different architectures [FORT 1985, STON 1987, BOKH 1988] and divided in to three classes: array computers, pipeline computers and multiprocessor systems. To name a few are IBM 3081, Denelcor HEP, Cray x-MP and PARAM systems. All the parallel computers described above are centralized computing items and all hardware and software resources are housed at the same place.

1.3.6. Types of Communication Systems

1.3.6.1. Horizontal Vs Vertical Distribution

By Vertical distribution we mean that there is a hierarchy of processors, shown in Fig.1.1 [MART 1988]. The transaction may enter and leave the computer system at the lowest level. The lowest level may be able to process the transaction or may execute certain functions and pass it up to the next level. Some, or all, transactions eventually reach the highest level, which will probably have access to on-line files or databases. The computers at the lowest level can be networked together, if data sharing is required.

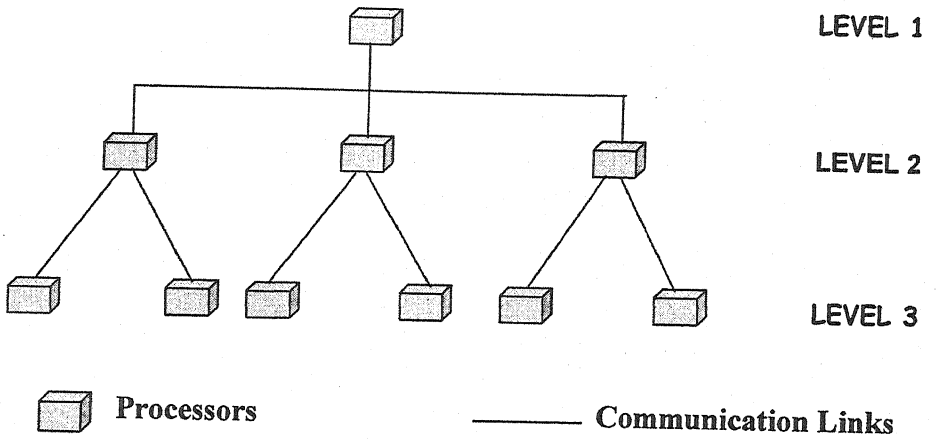


Figure 1.1 Vertical Distributions

A horizontal distribution implies that the distributed processors do not differ in rank. They are of equal status-peers and referred as peer-coupled systems. A transaction may use only one processor, although there are multiple processors available. On some peer-coupled systems a transaction may pass from one system to another causing different set of files to be updated as depicted in Fig.1.2 [MART 1988].

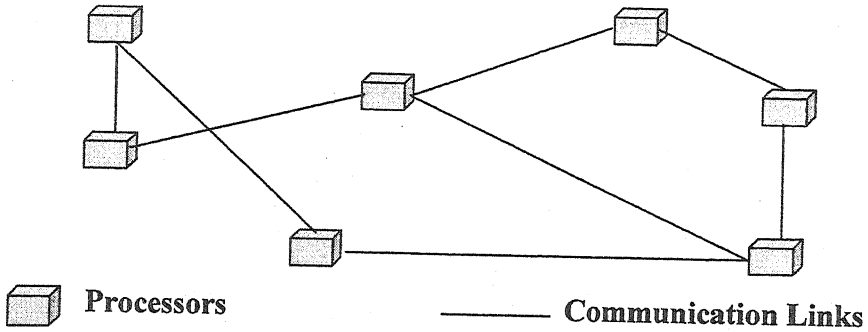


Figure 1.2 Horizontal Distributions

1.3.6.2. Functional Distribution Vs System Distribution

In some distributed systems, usually vertical systems, functions are distributed, but not the capability to fully process entire transactions. The lower-level machines may be intelligent terminals or intelligent controllers in which processors are used for functions such as message editing, screen formatting, data collection dialogue with terminal operators, security, or message compaction concentration. They do not complete the processing of entire transactions.

This distribution is referred as functional distribution and is contrasted with system distribution in which the lower-level machines are systems their own right, processing their own transactions, and occasionally passing transactions or data up the hierarchy to higher level machines.

In a system distribution environment the lower machines may be entirely different from, and incompatible with, the higher machines. In a function distribution environment, close cooperation between the lower-level and higher-level machines is vital. Overall system standards are necessary to govern what functions are distributed and exactly how the lower and higher machines form part common system architecture with appropriately integrated control mechanisms and software.

1.3.7. Feature of Communication System

There are a variety of reasons for building any communication System, some of them are as follows:

(i) **Computational Speedup:** Computational speedup can be achieved if a particular computation can be partitioned into a number of sub-computations that can run concurrently. A distributed system may allow user to distribute the computation among the various sites – to run that computation concurrently. In addition, if a particular site is currently overloaded with the jobs, some of them may be moved to other, lightly loaded, sites.

(ii) **Fault Tolerance:** One of the real attractions of distributed processing is the resulting high fault tolerance. If a communication link or a site fails in a distributed system, the remaining sites can potentially continue operating. However, this may reduce the overall throughput of the system.

(iii) **Increased Throughput:** The throughput of the system is expected to increase by distributing the total workload to the various service stations. As in a distributed system there are a number of processing elements, one would hope to get more work done in shorter period of time.

(iii) **Communication:** In the distributed system it is quite often the programs running at different sites need to exchange data with one another. When a number of site are connected to one another by a communication network, the processes at different sites have the opportunity to exchange the information.

(iv) **Resource Sharing:** If a number of sites are connected to one another, then a user at one site may be able to use the resources available at another. In general, resource sharing in a distributed system provides mechanisms for sharing files at remote sites, processing information in a distributed database, printing files at remote sites, utilizing specialized hardware devices, and performing other operations.

1.3.8. Application area of Communication Systems

Distributed processing plays an important role in large data base installations where processing load is distributed for organizational efficiencies. Banking system, travel agency systems, and power control systems are few examples of distributed processing environment. The application that exhibits parallelism, involving enormous repetitive processing and/ or requiring extremely fast processing in a real-time environment demand Communication Systems. To name a few such type of application

are signal processing, meteorology, image analysis, cryptography, nuclear reactor control, solar & radar surveillance, simulation of VLSI circuits, and industrial process monitoring.

1.4 TASK ALLOCATION

The task allocation in a communication system finds extensive applications in the faculties where large amount of data is to be processed in relatively short period of time, or where real-time computations are required. Meteorology, Cryptography, Image Analysis, Signal Processing, Solar and Radar Surveillance, Simulation of, VLSI circuits and Industrial process monitoring are areas of such applications. These applications require not only very fast computation speeds but also different strategies involving distributed task allocation systems. In such applications the quality of the output is proportional to the amount of real-time computations.

To meet such challenging computing requirements at electrifying speeds some efficient task allocation strategies are required for proper utilization of such system under the constraints as:

(i) Memory capacity available at each processor and

(ii) Time constraints in executing task.

The advent of VLSI technology resulting in low cost microprocessor has made CS an economic reality in today's computing environment. The modularity, flexibility and reliability of CS make it attractive to much type of users, and several CSs have been designed and implemented in recent years.

The first step in the distributed software engineering is to partition application program into a set of smaller independent tasks and allocates them to different processors. To enable the increasing variety of computers to communicate with one another, there must be rigorously defined protocols as to (i) how the Control Message [CM] and Data Message [DM] are exchanged in the CS and (ii) how to control the communication process along with the protocol definition.

The format of the CMs, the headers and the trailers of DMs are likewise rigorously defined. Protocol becomes quite complex, as it is desirable that there should be a widely accepted standard so that all types of machines can inter communicate. However, many problematic areas in the CSs are still in their primitive development stage. Some major problems present the following widespread uses CS are:

(i) The degradation in system throughput caused by the saturation effect,

(ii) The difficulty in evenly utilizing each processor in the CS,

(iii) A large gap between the engineering application requirements and existing distributed network architecture and

(iv) The difficulty in verifying task allocation resulted from any allocating model.

Communication Systems have been so complex that intuition alone is not sufficient to predict their performance. Therefore, mathematical modeling plays an important role for predicting the performance of the CS. Mathematical models of system performance range from relatively simple ones, whose solution can be obtained analytically, to the complex ones, which require simulation.

Assigning tasks to processors is called task allocation, which involves the allocation of tasks to processors in such a way that some effectiveness measures are optimized. If the effectiveness measure can be represented as a linear function of several variables subjected to a number of linear constraints involving these variables, then the task allocation is classified as a linear programming problem. Likewise, for the processor, which can perform anyone the several tasks, possibly the difference of execution, and the effectiveness measure is the total processing time to perform all tasks when one and only one task is allocated to each processor. In such cases, task allocation is classified as an assignment problem. Assigning "m" tasks to "n" processors, through exhaustive enumeration, results in n^m possible ways. A general structure of task allocation is shown in Fig.1.3.

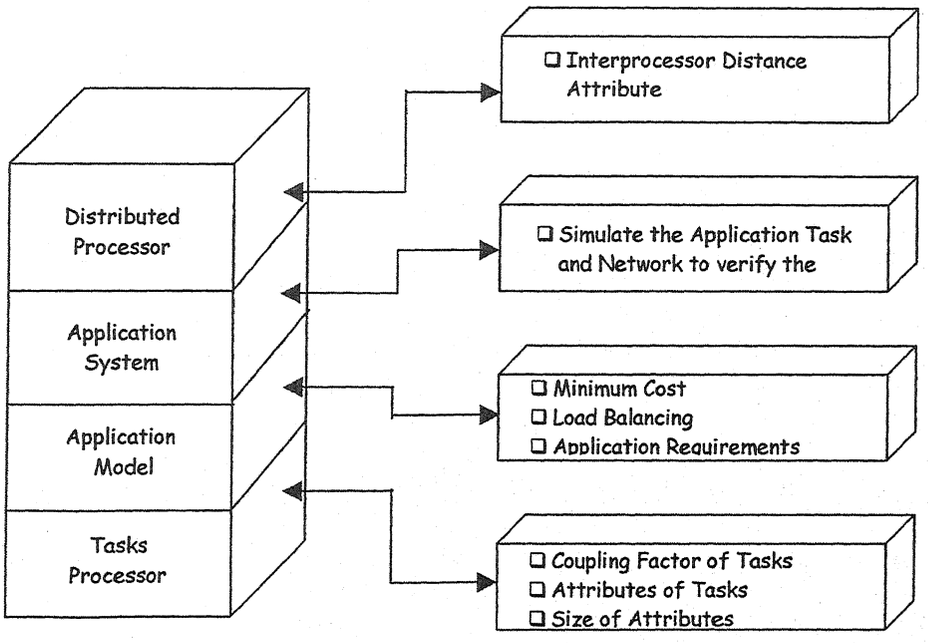


Figure 1.3: General Structure of Tasks Allocation

Shatz and Wang [SHAT 1987] studied that the problem of choosing an Optimal Allocation from all assignments is exponentially complex. An efficient task Allocation policy should avoid excessive Inter Processor Communication [IPC] and exploit the specific efficiencies of the processors and in case of a system having similar processor, the tasks or modules should be distributed as evenly as possible. The bottleneck in IPC is to provide linear speed up solutions with the increase in number of processors as suggested by Chu et al and Lint et al. [CHU 1980a, LINT 1981].

The strategies of task allocation on a parallel and distributed system may be done in any of the following ways:

(i) **Static Allocation:** In static allocation when a task is assigned to processor, it remains there while the characteristic of the computation change, a new assignment must be computed. The phrase **"characteristics of the computation"** means the ratios of the times that a program spends in different parts of the program. Thus in a static allocation, one is interested in finding the assignment pattern that holds for the life time of a program, and result in the optimum value of the measure of effectiveness.

(ii) **Dynamic Allocation:** In order to make the best use of resources in a distributed system, it is essential to reassign modules or tasks dynamically during program execution, so as to the advantage of changes in the local reference patterns of the program [MANI 1998]. Although the dynamic allocation has potential performance advantages, Static allocation is easier to realize and less complex to operate.

1.4.1 Task Allocation Problem

Consider a set P= {p_1, p_2, p_3,……., p_n}, of "n" processors interconnected by communication links and a set T= {t_1, t_2, t_3,……., t_m}of "m" executable tasks. The allocation of each "m" tasks to "n" available processors such that an objective cost function is minimized subject to the certain resource limitations and constraints imposed by the application or environment.

An assignment of tasks to processors is defined by a function f, from the set of tasks T to the set of processors P, f: T→ P. Then the total cost or Time of an allocation x can be expressed as follows:

$$Cost(x) = \sum_{i=1}^{m}\sum_{j=1}^{n}\left[e_{ij}x_{ij} + \sum_{k<i}\sum_{1<j} c_{ik}d_{jl}x_{ij}x_{kl}\right]$$

where,

$$x_{ij} = \begin{cases} 1, & \text{if } i^{th} \text{ task is assigned to } j^{th} \text{ processor} \\ 0, & \text{otherwise} \end{cases}$$

e_{ij} = cost of executing task t_i on processor p_j

c_{ik} = Communication between tasks t_i and t_k

d_{jl} = distance between processors p_j and p_l

1.5 SOME RELATED APPROACHES

Researchers have identified several approaches to the task allocation models. The following of them are concern with our work:

1.5.1 Graph Theoretical Approaches

The graph theoretical approaches suggest graphical methods to represent and allocate application tasks to various processors in a communication system. In most of the graph theoretical approaches, the solution processes begins with the abstraction of tasks and inter task communication cost through a graph model in which tasks are represented by nodes and inter task communications as weights on the non-directed edges connecting these nodes. By performing the min-cut algorithm the graph minimizes the total computation cost. These approaches do consider load balancing, resource limitation, and generalization issues without giving much consideration to the computing time and complexity. Graph theoretical algorithms become NP-hard for CS consisting of reasonable number of processors.

1.5.2 The Integer Programming Approaches

The integer programming approaches are based on the direct enumeration algorithms subject to the additional constraints. This allows real time embarrassments, and rather complicates it to be incorporated into the allocation model to meet various application requirements. These approaches are limited and do consider the system

resources limitation, the amount of real time computation, and memory needed to obtain an optimal solution, as these grow exponential functions of the order of the problem.

1.5.3 The Heuristic Approaches

The heuristic approaches provide approximate solutions to task allocation and load balancing problems. These approaches require less computation time as compared to the integer programming methods and are applicable in the situations where an optimum solution is not achievable within the reasonable time limit.

1.6 INTRODUCTION TO RELIABILITY

Reliability theory has grown into an engineering science in its own right following the World War II. Much of the initial theory, engineering, and management techniques centered about hardware, however, human and procedural elements of a system were often included. Since the late 1960s the term software reliability has become popular, and now reliability theory refers to both software and hardware reliability [SHOO 1983].

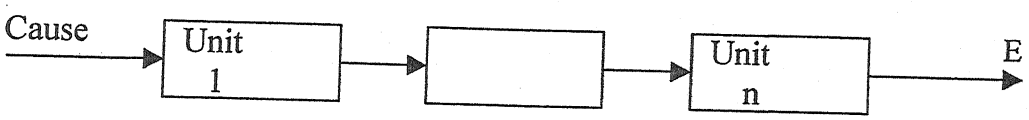
1.6.1 Combinational Reliability

In performing the reliability analysis of a complex system, it is almost impossible to treat the system in its entirety. The logical approach is to decompose the system into functional entities composed of units, sub systems, or components. Each entity is assumed to have two states, one operational and one failed. The sub division generates a block-diagram or fault-tree description of system operation. Models are then formulated

to fit this logical structure, and the calculus of probability is used to compute the system reliability in terms of the sub division reliabilities.

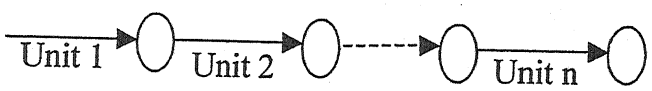
1.6.2 Series Configuration

The simplest and perhaps the most common structure in reliability analysis is the series configuration. In the series case the functional operation of the system depends on the proper operation of all system components. A series reliability configuration is portrayed by the block-diagram representation shown in Figure 1.4(a) and the reliability graph shown in Figure 1.4(b). In either case, a single path from cause to effect is created. Failure of any component is represented by removal of the component, which interrupts the path thereby causes the system to fail.



Block Diagram of Series with n-components

Figure 1.4(a)

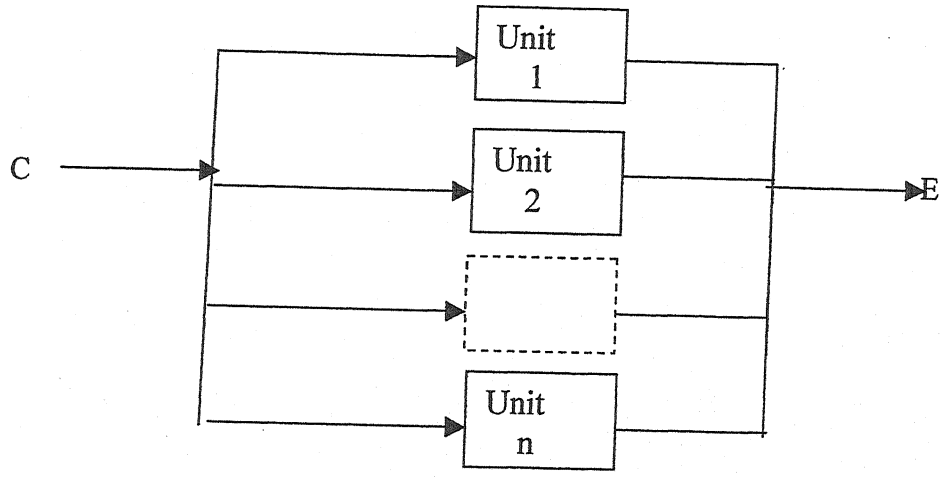


Reliability Graph of a Series system with n-components

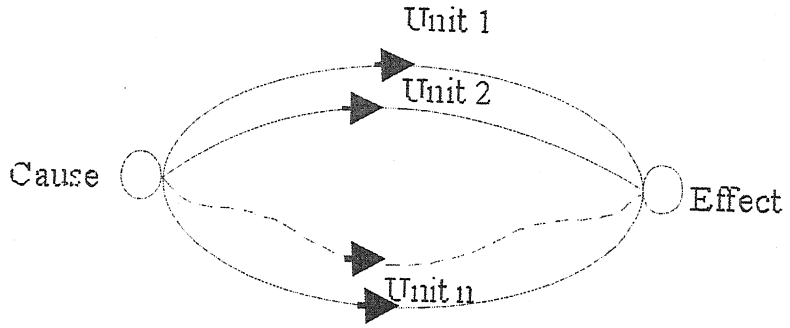
Figure 1.4 (b)

1.6.3 Parallel Configuration

In many systems several signal paths perform the same operation. If the system configuration is such that failure of one or more paths still allows the remaining path or paths to perform properly, the system can be represented by a parallel model. Block diagram and reliability graphs for a parallel system are shown in Figures 1.5(a) and 1.5(b) respectively. There are "n" paths connecting input to output, and all units fail; in order to interrupt all the paths. This is sometimes called a redundant configuration.



Block Diagram of Parallel System with n-components
Figure 1.5 (a)



Reliability Graph of a Parallel System with n- Components
Figure 1.5 (b)

1.6.4 An-r-Out-of-n Configuration

In many problems the system operates if "r" out of "n" units function e.g., a bridge supported by "n" cables, "r" of which are necessary to support the maximum load. One can draw a reliability graph as an aid. Each parallel path will contain "r" different elements, corresponding to the combinations of "n" things "r" at a time.

1.7 SCHEDULING POLICIES IN A CS

The scheduling policies in a CS have two categories such as job scheduling and task scheduling. Job scheduling allocates independent jobs to different processors to optimize system performance. Task allocation scheduling requires assignment of multiple interdependent tasks or modules of a single allocation program to minimize job completion time. The decision of scheduling polices is required at several levels within CS as shown in Fig. 1.6. At higher level the decision about local or global scheduling might be desired to make. The local scheduling policies decide about the allocation of a task or a job to the time slots for a single processor [BUCK 1979, CASA 1988]. The global scheduling relates to the problem of deciding where jobs or tasks [ROTI 1994]

run. In the next lower level is the choice between static and dynamic in a global scheduling. Static scheduling is prior assignments tasks to the processors where the allocation does not change during the lifetime of tasks. While in dynamic scheduling the allocation decision is made during execution of tasks. Further there are two type of dynamic scheduling: centralized and decentralized. If there is a single decision point, the dynamic scheduling is known as centralized and if the decision-making is spread throughout the system it is known as de-centralized.

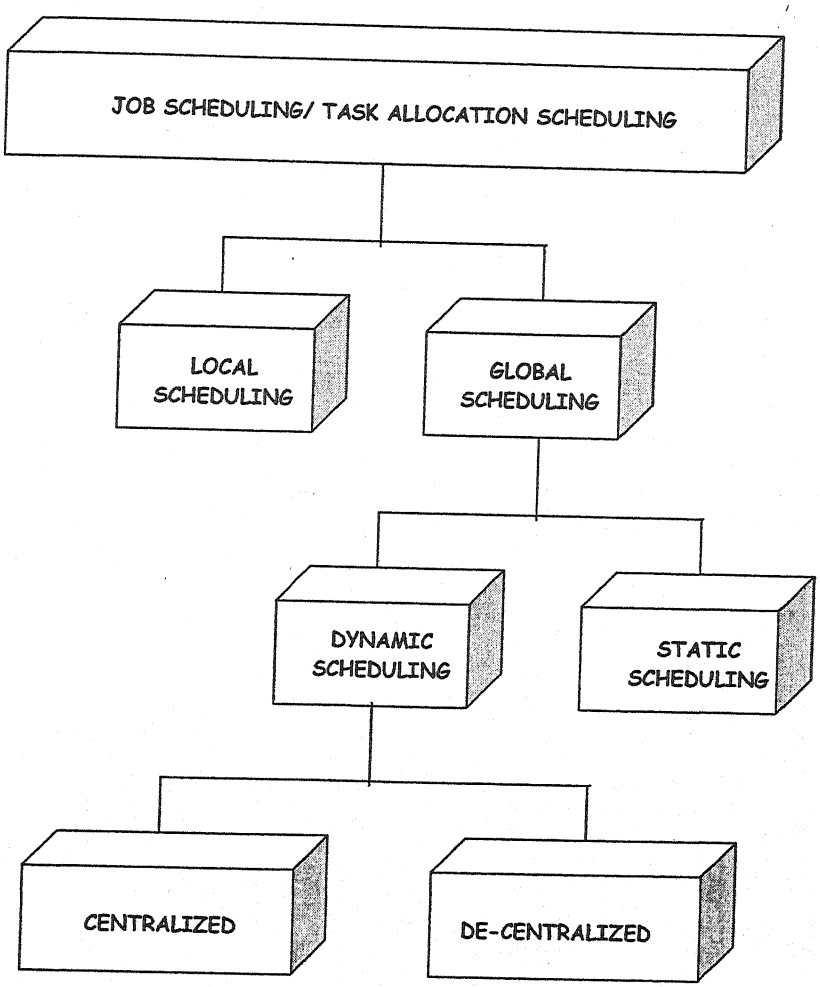


Figure 1.6: Taxonomy of Scheduling

1.8. DEFINITIONS, ASSUMPTIONS AND NOTATIONS

1.8.1 Definitions

(i) **Inter Task communication Time:** The Inter Task Communication Time t_{ik} of the interacting task's t_i and t_k depends on the period in which data units exchanged between them during the time execution.

(ii) **Execution Time:** Each task t_i has an Execution Time et_{ij} (where $1 \leq i \leq m$ and $1 \leq j \leq n$) that depends on the processor P_j to which it is assigned and the duration of work to be formed by the task.

(iii) **Load Balancing:** Spreading the total load of a program / application on the available processing elements as evenly as possible is referred to as load balancing.

1.8.2. Assumptions

To cope up with the real life problems and to keep the algorithms reasonable in size the following assumptions are made:

(i) The number of tasks to be allocated is more than the number of processors, as normally is the case, in the real life CSs.

(ii) Whenever two or more tasks are assigned to the same processor the Inter Tasks Communication Time between them is assumed to be zero.

(iii) Whenever two or more tasks are assigned to the Inter Processor Distance between them is assumed to be zero.

(iv) If a task is not executable on a certain processor due to absence of some resources, the Execution Time of the task is taken to be infinite (∞).

1.8.3 Notations

CT	:	Communication Time
CTM(,)	:	Communication Time Matrix of Order m x m
et_{ij}	:	Time of execution of the i^{th} task on the j^{th} processor
CRM(,)	:	Communication Reliability matrix of order m x m
CR	:	Communication Reliability
CUR	:	Communication Unreliability
CURM(,)	:	Communication unreliability Matrix of order m x m
ET	:	Execution Time
ETM(,)	:	Execution Time Matrix of order m x n
EUR	:	Execution Unreliability
EURM(,)	:	Execution Unreliability Matrix of order m x m
IPROD	:	Total number of assignments (Combinations)
m	:	Number of Tasks
n	:	Number of Processors
MCT(,)	:	An array to store Maximum Communication Time
nar	:	Counter for Allocation
NETM(,)	:	Modified Execution Time Matrix of order n x n
NERM(,)	:	Modified Execution Reliability matrix of order n x n
NEURM(,)	:	Modified Execution Unreliability Matrix of order n x n
P	:	Set of Processors $\{p_1, p_2, p_3, \ldots\ldots\ldots\ldots\ldots, p_n\}$
PCOMB	:	An Array to store processors combinations
POS[,]	:	A matrix to sore the positions of communication time on the bases of MCT( ).
T	:	Set of task $\{t_1, t_2, \ldots\ldots\ldots t_m\}$
T_{ass}	:	Assigned Tasks
TCOMB( )	:	An Array to store tasks combinations
$T_{non\text{-}ass}$	:	Non-Assigned Tasks
NCTM (,)	:	Modified Communication Time Matrix of order n x n
pet ( )	:	An array to store processor wise execution Time
itctm(,)	:	Inter tasks Communication Time Matrix of order m x n

ipdm(,)	:	Inter tasks processors distance Matrix of order n x n
ITCT	:	Inter tasks Communication time
ITCTPD	:	Inter task communication time with processor distance
itctpd( )	:	An array to store processor wise communication time with Processor distance
pct ( )	:	An array to store inter processor communication time
TET	:	Task Execution Time
TETM (,)	:	Task Execution Time matrix of order m x n
TWM ( )	:	Task Weight Matrix of order m

1.9 ORGANIZATION OF THE THESIS

The First chapter of this thesis entitled "TO STUDY AND ANALYSIS OF OPTIMIZATION TECHNIQUES AND THEIR APPLICATION TO EVOLVE SOME ALGORITHM TO PERFORMANCE ENHANCEMENT OF COMMUNICATION SYSTEM" is devoted to the introductory background of the topic in optimization techniques, communication systems, task allocation and other issues directly related to the work. A sequential historical introduction to the optimization has been described in details. Various aspects of the optimization have been mentioned. This chapter contains the different techniques of the optimization such as, Graph theoretic approaches, Integer programming approaches and many more. A brief overview of the communication system has been given to understand the basics of it. Assignment problem or let us say task assignment problem in these systems, is discussed and also some of the related concept has been included. In order to complete this piece of research we use number of definitions, assumptions and notations, all such definitions, assumptions and notations, which are used throughout the thesis has been put together and listed in this chapter.

Finally, the organization of the thesis, include the chapter wise brief summary of the present work has been mentioned in this chapter.

Task assignment of any communication system is a most computing interesting and demandable research problem. Various methodologies and techniques are available in the literature to solve such problems. As it is the primary for researchers to known about related methodology and techniques.

The Second Chapter of this thesis is a collection of all such research work, which is available in the literature, and directly-indirectly correlate to our work. We have categories them in two groups such as commonly adopted approaches and consideration of a particular issue. In commonly adopted approaches we have considered the Branch and Bound methods, Dynamic Programming, Decomposition approach, Genetic algorithmic approach, Heuristic approaches, Integer Programming approaches, Network Partitioning algorithm, Problem Reduction method, Petri Net modeling & Reliability evaluation and Shortest Path algorithmic method. On the other hand for consideration of a particular issue, we have chosen the precedence constraints, particular architecture consideration and reliability evaluation.

The Third Chapter of the thesis is the important and main, as it include all five research problems. The general Motivation for the present research has been written briefly. The Objective, Technique, Computational Algorithm, Implementation and

Conclusion of each problem are mentioned in this chapter. The Details of each problem is as mentioned below:

In the Problem – I, the assignment problems in communication systems are one of major factor to determine the performance of such systems. The present piece of research problem suggests an exhaustive search approach to obtain the optimal time for optimally assigning the tasks to the processors of communication system. Here we define an index that is based on the time for the execution of task to various processors and also communication time amongst the tasks. The computational algorithm of the approach and its implementation are mentioned. The results are shown in tabular form as well as graphical form. This study is capable to deal all such real life situations, where the tasks are more than the number of processors. The developed algorithm is coded into C++ and several sets of data have been tested to verify the effectiveness of the algorithm.

The title of the Problem – II is Optimizing the time of CS: Matrix Partitioning Approach. The performance analysis of any communication system has an important area of research these days. The assignment problems in communication systems are one of the major problems to determine the performance of these systems. The present piece of research problem – II suggests a matrix partitioning approach to assign all the tasks to the available processors of the system. Here we have chosen such a problem in which the tasks are more as compared to the processors of the system. It contains the computational algorithm of the technique and its implementation. This piece of research is capable to deal all such real life situations, where the tasks, which are to be executed on the

processors of the communication systems, are more than the number of processors of that system. This technique does not require adding dummy processors. The developed algorithm is coded into C++. The several sets of input data have been tested to verify the effectiveness of the algorithm.

In the Problem - III, a new approach for assignment in communication system based on tasks weight matrix partitioning has been considered. This suggests an efficient algorithm for communication system. The problem chosen, in which the number of tasks is more than the number of processors in the system, as it is most of the time the case of real life situation. The tasks are allocated to the processor in such a way that to suit the desired execution requirement of the tasks. The model addressed here is based on the consideration of task execution time and task weights. The method is presented in algorithmic form and implemented on the several sets of input data to test the performance and effectiveness of the algorithm. The developed algorithm is coded into C++.

In the Problem - IV, we have used the Matrix partitioning technique to improve the reliability of communication system. It is almost impossible that the communication systems has to execute only as many tasks as the number of processors available in the system. That means the number of tasks, which are to be executed on the communication systems, shall be the more as compared to the number of processors in the systems. The problem of execution of "m" tasks to "n" processors ($m > n$) in a communication systems is addressed here through an efficient algorithm for the task execution in a

communication system. The execution reliabilities of the tasks on different processors have taken into consideration while preparing the algorithm to such a case. The model discussed here is based on the consideration of execution unreliability's of the tasks to the processors. Keeping in view we suggested an efficient method to assign all the tasks as per the required availability of processors so that none of the tasks get remains unexecuted and the present approach does not require to adding dummy processors. The several sets of input data are considered to test the effectiveness and efficiency of the algorithm. It is found that the algorithm is suitable for arbitrary number of processor with the random program structure and workable in all the cases.

Our day-to-day experience reveals that processor distance influences the overall task completion cost in any communication system, for instance, in the case of telecommunication network. It is well known fact that a person has to pay more for talking to person sitting at a long distant place. The case of Communication System is similar to that of a telecommunication network in this aspect. In a system, a task is allocated to a processor in such a way that extensive Inter Task Communication is avoided and the capabilities of the processor suit to the execution requirements of the task.

The model discussed in the Problem - V provide an optimal solution for assigning a set of "m" tasks of a program to a set of "n" processors where $m > n$ in a Communication System with the goal to maximize the overall throughput of the system and allocated load on all the processors should be balanced. The present allocation policy

involves stepwise modification of execution and communication matrices by making the clusters tasks. Some of the tasks may not involve in any cluster, those tasks treated as independent tasks. The mathematical programming approach has been used to determine the optimal allocation of tasks. The several sets of input data are considered to test the complexity and efficiency of the algorithm. It is found that the algorithm is suitable for arbitrary number of processor with the random program structure and workable in all the cases.

In the Fourth Chapter, as we all know that research is an on going process and there is no last limit so that we have coin some of the new research problems, each are to be solved by the future researcher. Keeping these things in mind, the last chapter of this thesis contains the future scope of this research area and new research problem along with the suggestions for the researchers.

During the period of this research work some of the research papers have been presented in various seminar and conferences. Two research papers are published in Journal (Copies of both papers are appended at the end of the thesis). And also some of research papers have been sent for the publication in various journals. A combined list of such papers has been mentioned in this chapter. During this research work we have gone through large number of research papers, books, monographs, Ph. D. thesis etc. so that in the last of the thesis a list of in alphabetic order of such research material has been attached.

CHAPTER-2

SURVEY OF THE LITERATURE

2.1 AN OVERVIEW

Over the past few years a number of studies in optimization techniques and their applications to Communication System (CS) have led to the identification of several challenging problems. One of these problems is optimally assigning the tasks to the processors in the CS. In such system, it is essential to assign each task to the processor whose characteristic are more suitable for the execution of the task. Availability of information at geographically dispersed locations and the need of its transmission from one source to another have also led to the development of CS.

The task allocation problem in a CS may either be static or dynamic. If the allocation is static the information regarding over all costs of a hardware configuration is known before hand. When a task is assign to a particular processor, it stays on that processor during its life time. The problem of assigning m tasks to n processors, where m > n as normally is the case, in the real life is a complex one. The complexity may be reduced by an efficient task partitioning strategy [BERG 1987]. In case of dynamic allocation the tasks or modules can reassign dynamically during program execution, so as to the advantage of changes in the local reference patterns of the program [MANI 1998]. Although the dynamic allocation has potential performance advantages, static allocation

is easier to realize and less complex to operate. Recently, Zhou, Zhibo, Zhou, Tong and Jinxiang Wang [ZHOU 2007] have discussed the bit error rate performance of an Invariant Delay Multiple Access DCSK (IDMA-DCSK) Communication System over multi path fading channels. They have established the close relation between theoretical and the simulation results. Ha, C and Kuo, W. [HA 2006] has also presented the multi-path heuristic for redundancy allocation. Onishi, J., Kimura, S., James, R.J.W., Nakagawa, Y., [ONIS 2007] has solved the redundancy allocation problem by using surrogate constraint method. A through survey of the related literature reveals that the earlier researches can be classified in the following two major categories

.

- Commonly Adopted Approaches
- Consideration of a Particular Issue

2.2 COMMONLY ADOPTED APPROACHES

2.2.1 Branch –and – Bound Methods

Branch – and –Bound technique is very popular and widely adopted technique among the researchers. There exist several algorithms based on this technique for task allocation problem [AROR 1980, MA 1984, SINC 1987]. An optimal task allocation strategy with main concern of maximizing reliability and considering communication cost as the constraint function for a distributed database management system has been reported in [VERM 1997]. The model is converted into a state space search tree and Branch- and –Bound technique is used to achieve the optimum results. This approach has

been adopted for optimizing the distributed execution of join queries by Reid [REID 1994].

2.2.2 Dynamic Programming

Dynamic programming is an algorithm design method that can be used when the solution to a problem may be viewed as the result of a sequence of decisions. The main feature of this technique is that it often drastically reduces the amount of enumeration by avoiding the enumeration of some decision sequences that can not possibly be optimal. This approach aims at finding the optimal sequence of decision employing the principal of optimality. Dynamic programming based solution procedure can be regarded as recursive optimization because the decisions, leading to an optimal solution, are made one at a time. A number of research papers based on dynamic programming approach are available in the literature of task allocation problem in CS. Employing this approach, Bokhari devised a shortest tree algorithm that runs in $o(mn^2)$ time for the assignment of `m' modules to `n' processors [BOKH 1981a]. A relationship between module allocation and non – serial dynamic programming has been established by Rosenthal [ROSE 1982]. Considering cost and reliability aspects, Ashrafi et al [ASHR 1992] developed optimization models for selection of programs. Dynamic programming technique is applied by Ferman Dez-Baca [FERN 1989] for obtaining the optimal assignment through local search.

2.2.3 Decomposition Approach

The reliability evaluation of complex networks using decomposition method is also one of the important techniques. The reliability of an oriented type-I network is

computed by Nakazawa [NAKA 1976] by decomposing the network into short-and-open-removed sub networks according to its keystone line called a removable line. In an another paper [NAKA 1977] the type-I of networks is extended to types-II, II & III and equivalence of one non oriented line and one pair of oriented lines with equal 1-reliability is proved for types I, II & III. It is also proved that the type III networks have to be transformed into a Boolean expression and then the Boolean decomposition method [NAKA 1977] can be applied. A network [NAKA 1978] is represented by matrices which is suitable for computer use. The criteria for removable lines of complex networks are proved. It is also proved that there are always removable lines of complex networks except oriented type III one, using the matrix expression [NAKA 1981]. For determining the conditional success events Aggarwal et al [AGGA 1982] evolved a technique using both the node removal and connection multiplication methods for path enumeration. Li et al [LI 1992] suggested an algorithm for optimization of large-system reliability with the help of decomposition method.

2.2.4 Genetic Algorithmic Approach

This optimization approach is a heuristic optimization technique, which includes simulated annealing. Tabu searched an evolutionary strategy [REEV 1993]. Coit and Smith [COIT 1996] developed a specific genetic algorithm to analyze series-paralleled system and to determine the optimal design configuration where redundancy allocation problem for a series-parallel system had a specified number of sub systems and for each sub system, there are multiple component choices which can be selected and used in parallel. For those system designed using off- the- shelf component types, with known cost, reliability and weight, the system design and component selection become a

combinational optimization problem. They [COIT 1998a] also presented an algorithm to solve the redundancy allocation problem when the objective is to maximize a lower percentile of the system time- to- failure distribution. Levitin et al [LEVI 1998] discussed a redundancy optimization problem to multi-static systems. Here the systems and their components have the range of performance levels. Dengiz et al [DENG 1997] used the similar approach to solve the all-terminal network design problem when considering cost and reliability. Genetic algorithm was also used in optimization of system reliability [KUMA 1995b, PAIN 1995], while Kumar et al [KUMA 1995a] worked on the topological design for such systems. Exploring Genetic Programming and Boosting Techniques to Model Software Reliability has been studied by Costa and E.O. de Souza, G.A., Pozo, A.T.R., Vergilio, S.R. [COST 2007].

2.2.5 Heuristic Approach

Heuristic Algorithm has also played a vital role for providing solutions to different types of problems. Several research papers [DENG 1997, LU 1986, WARD 1984, PRIC 1984, WILL 1983, ALFO 1988, HWAN 1993, EFE 1982, AROR 1980] have been reported in the literature regarding heuristic approaches for task allocation problem of these systems. The use of CS has been rapidly increasing in order to share expensive hardware and software resource and provides access to main systems from the distant locations. Dengiz et al [DENG 1997] suggested a heuristic algorithm inspired by evolutionary approach to solve the all terminal network design problem when considering cost and reliability. Kartik and Murthy [KART 1995] provided a heuristic algorithm to find an optimal and sub optimal task allocation in redundant distributed computing

systems, to maximizing system reliability. Further it repeatedly consider all the possible reassignments of tasks at a time, performs the most advantages reassignments and terminals when no more profile reassignments is discovered. On the other hand, the problem of minimizing the cost by making the clusters of those tasks that is heavily communicating and assigning the tasks cluster to appropriate processors. On the basis of execution drew considerable attention [WILL 1983, PRIC 1984, WARD 1984]. Some heuristic approaches for task scheduling based on characterization of inter module communications [LAN 1985] and reliability maximization [HWAN 1993] in the distributed computing system also deserves mention. The heuristic load balancing approaches [BOLC 1983, YANG 1987] suggested reducing the number of tasks by forming task clusters equal to the number of processors. This reduction is accomplished by fusing those tasks between them data exchanges are maximum. The task-clusters so formed are assigned to the processors. Finally tasks are shifted from the heavily loaded processors to the minimally loaded processors to balance the load on all the processors. Liu X. et al [LIU 1999] has presented a study of building systems from components to answer the questions related the configuration of components and performance relation with monolithic system. They has also discussed the generality of the technique with regard to systems other than CS. Wang, N, Lu, J.C. and Kvam, P [WANG 2007] has also given the reliability modeling in spatially distributed logistics systems. Another simple heuristic algorithm for generating all minimal paths has been suggested by Yen Wei Chang [YEN 2007].

2.2.6 Integer Programming Approach

A task allocation model becomes more significant and realistic when it incorporates real- time constraints such as inter-processor communication, memory limitations of each processor, etc. For the task allocation problems, Integer Programming is a useful and exhaustive approach as it is capable to reflect real- life situations of distributed processing and simplicity in application. A number of researchers have worked on task allocation problems employing this technique to determine the optimum solution [MA 1982, DESS 1980, CHU 1980a]. A model based on this approach is developed by Chu for optimum file allocation in a multiple computer system [CHU 1969]. Another similar approach for data file allocation has been reported by R. Marcogliese and Novarese [MARC 1981].

2.2.7 Network Flow and Network Partitioning Algorithm

Network Flow algorithm keeps an important role for the task allocation problems. Stone used this technique for the task allocation in the multiprocessor systems in 1977 [STON 1977] by making use of Ford Fulkersen algorithm [FORD 1962] for finding maximum flows in commodity networks as modified by Edmonds and Karp [EDMO 1972, KARZ 1974, DINI 1970]. The complexity of this algorithm is known to be O $[(m+2)^3]$, where m is the number of tasks. This method provides no mechanism for representing memory size or other constraints. Wu et al [WU 1980a] employed network flow algorithm for task allocation and also described the partition algorithm for parallel and distributed processing [WU 1980b]. In another algorithm load balancing in a circuit-switched multi computer has been used by Bokhari [BOKH 1993]. Ke and Wang [KE

1997] proposed an efficient reliability evaluation algorithm accounting for imperfect nodes in distributed computing networks. Based on the concept of network partition, the algorithm exploits some simple efficient techniques to handle the unreliable nodes, for directly computing the network reliability expression considering imperfect nodes instead of using any compensating method.

2.2.8 Problem Reduction Method

A smaller problem can be handled more effectively in comparison to a bigger problem. It is therefore, always advisable to reduce the size of a given problem, if possible. A task allocation problem can be simplified to a certain extent by reducing the size of the problem. Arora and Rana, who proposed module assignment in two processor distributed system [AROR 1979], adopted this idea for proposing the concept of clustering the tasks which exhibit certain peculiar behavior to reduce the problem size [AROR 1981]. A task is either assigns to a processor or fused with another task(s) depending upon some criterion. Sagar and Sarje [SAGA 1991] used the above technique for task allocation model for evolving new distributed systems. A clustering algorithm for assignment problem of arbitrary process systems to heterogeneous distributed computer system is implemented by Bowen et al [BOWE 1992]. He showed that clustering algorithm exhibit better time complexity. The clustering technique used by Kim and Browne [KIM 1988] iteratively applies a critical path algorithm to transform the graph into a virtual architecture graph which consists of a set of linear clusters and the interconnection between them. A new heuristic, based on the concept of clustering, to allocate tasks for maximizing reliability is proposed by [SRIN 1999]. Based on clustering approach, a new multiprocessor scheduling technique named de-clustering has been

given by Sih and Lee [SIH 1993b]. They claimed that their de-clustering approach not only retains the clustering advantages but at the same time overcomes its drawbacks.

2.2.9 Petri Net Modeling & Reliability Evaluation

The probabilistic graphs, being static in nature can not efficient model and analyze the dynamic behavior of the system [JOLL 1980]. A Petri net approach introduced by Petri [PETR 1962] and Holt [HOLT 1978] is recognized as one of the most powerful modeling and analysis tool for a communicating processing environment. Petri net theory to model a distributed processing environment is also studied by Murata [MURA 1979]. Kumar and Aggarwal [KUMA 1993] suggested Petri net model for the evaluation of reliability for the execution of a computer program in a CS. The execution of a program in a distributed computing system may require access to several files residing at different sites and communications paths between several node pairs. Then, by using the ratability, firing and marking concepts of Petri net, an algorithm is developed to study the distributed computer program reliability and the CS reliability. Earlier Kumar et al [KUMA 1988, KUMA 1990] used this approach to obtain the reliability of a broadcasting network. Lopez [LOPE 1994] presented a model based on stochastic Petri net to estimate the reliability and availability of programs in a DPE. Some models [SHAT 1992, SHAT 1989] for reliability oriented task allocation in redundant distributed computer system and estimating the reliability and availability of programs in a distributed processing environment. Recently Schnee Weiss, W.G. [SCHN 2007] presented a review of Petri net picture book and Petri nets for reliability modeling. The reliability of tree structured grid services has been studies by Yuan – Shun, Dai [YUAN

2006, YUAN 2007]. The study related to confidence bounds for system reliability was presented by Ramirez-Marquiz, J.E. and Wei, Jian [RAMI 2006].

2.2.10 Shortest Path Algorithmic Method

The solution of the task allocation problem for a CS can be obtained by using the shortest path algorithms. Work based on such algorithms is reported in the literature [BOKH 1981a, PRIC 1982, TOWS 1986]. In order to apply this approach, a program graph is transformed into an assignment graph. For each node in the program graph, there exists a set of nodes corresponding to the number of processors in this assignment graph. The nodes and edges between them are labeled suitably depending on the method applied to obtain a shortest path from source- nodes to terminal- node. A shortest tree algorithm described in [BOKH 1981a] minimizes the sum of the execution and communication costs for arbitrarily connected distributed systems with arbitrarily number of processors, provided the interconnection pattern of modules form a tree. The objective of the study was to allocate the tasks, whenever possible, to the processors on which they execute fastest while taking into account the over head due to IPC. Dynamic programming approach was applied for determining the shortest path and then optimal assignment of the tasks of a program over the processors of an inhomogeneous DCS was obtained. The worst- case complexity of the method is found to be o $(n^3\phi^3)$, where, 'n' is number of nodes in assignment graph and 'ϕ' is the number of phases. Price and Pooch claimed that their shortest path method is applicable to all cases with the limitation that an optimal solution is possible in limited cases only [PRIC 1982]. The shortest path algorithm evaluates the set of nodes in the assignment graph corresponding to a program graph to

select the best possible allocation of a task to a processor. Towsley's work [TOWS 1986] is a special cases where the inter task communication pattern is series parallel in nature. A search is made in the assignment graph for the parts of the program graph where inter-task communication patterns are in series parallel or tree in nature. For such inter task communication patterns, shortest paths are derived and these shortest paths are combined to get the overall shortest path of the assignment graph. In most of the other related studies [BOKH 1987, IQBA 1986a, IQBA 1986b, KOHL 1975, BOKH 1988], and optimal task assignment graph is constructed which contains as many layers as number of processors. The calculation of weights of edges depends on the nature of the program graph and the processor mechanisms. Allocation of task interaction graphs to processor in heterogeneous networks is reported by Hui and Chanson [HUI 1997]. Keinghan [KEIN 1971] used a form of dynamic programming approach by partitioning a chain structured program graph. Bhardwaj et al. [BHAR 1994] have applied a similar method for optimal sequencing and arrangement in distributed single level tree networks with communication delay. Some bottlenecks of the shortest path methods are that the assignment graph becomes very large and complex as the number of processors in the distributed system increases and subsequently number of computations increase for calculating the edge weights. Further to add, the entire assignment graph has to be retained in the memory until the final assignment is made. Ramirez–Marquez, J.E. and Gebre, B.A., [RAMI 2007] suggested a classification tree based approach for the development of minimal cut and path vectors of a capacitated network.

2.3 CONSIDERATION OF A PARTICULAR ISSUE

2.3.1 Precedence Constraint

Earlier task allocation models did not consider modules precedence constraints as one of the significant factor. However, practically, it is not possible to start execution of a task before the completion of another task until the task is independent. Therefore, besides considering load balancing and minimization of IPC cost the precedence relation among task must also be taken into account. Some studies of task allocation models [CHOU 1986, MARK 1986] considered the precedence relation as a criterion for assignment. The model developed by Markencoff and Liaw [MARK 1986] is applicable for the problems exhibiting parallelism. The job comprising a set of tasks is modeled as a directed a cyclic graph. The graph has been partitioned into a set of sub graphs. Each directed a cyclic graph corresponds to an independent process without any data flow between sub graphs. The distribution of sub directed cyclic graphs, to processors, as evenly as possible, using branch and bound method, results in optimal assignment without IPC cost. The application for the response time of the threads is important rather than the entire application. Chu and Leung [CHU 1984b] reported an algorithm to search for task allocation that minimizes response time. The suggested algorithm considers the task replication and assignment together. Under any given task allocation process, to reduce response time, tasks reallocated from the longest weight processor to shortest weight processors until no further improvement is possible by such task reallocation. An extension for this algorithm with the objective of minimizing response time is proposed by Chu and Lan [CHU 1987b].The objective of task allocation is to counter balance the related but apposite factor of IPC cost and load balancing to achieve maximum throughput and to satisfy real time constraints.

2.3.2 Particular Architecture Consideration

The problem of assigning a set of 'm' tasks to a particular multiprocessor architecture having 'n' processors, where m > n, is known to be a mapping problem. This problem has been attempted by many researchers [BERM 1984, BERG 1985, BERG 1987, BOKH 1987, FUKU 1987, LEE 1987, MEND 1987, MURT 1988, MURT 1989, SIVA 1988, SIVA 1989, SHMI 1991, CHUA 1992, LIAN 1994, OLSO 1994] to obtain efficient mapping schemes. Chuang et al [CHUA 1992] suggested a fast recognition complete processor allocation strategy for hypercube computers. This model referred to a dynamic processor allocation scheme where the search space generated is dependent upon the dimension of the requested sub cube dynamically, rather than being predetermined and fixed. Olson et al [OLSO 1994] have evolved a model for the fault-tolerant routing in mesh architectures for the distributed computing system. Network flow models of message flow in the mesh and hypercube have been developed by Bokhari [BOKH 1993]. Sih and Lee [SIH 1993a] assumed the target architecture for their proposed scheduling technique to be shared- bus multiprocessor. Another polynomial-time algorithm is developed for computing the distributed program reliability for star topologies of CS in which data files are restricted to a certain type of distribution [LIN 1990].

2.3.3. Reliability Evaluation Consideration

Distributed computer systems are increasingly being employed for critical applications, such as aircraft control, industrial process control, and banking systems. Maximizing performance has been the conventional objective in the allocation of tasks

for such systems. There are many measures to evaluate the performances of CS. Reliability of a CS is another vary important issue that need special attention [AGGA 1982, GARC 1982, SEGE 1994]. A reliability measure called, distributed program reliability, to model accurately the reliability of CS has also been reported [KUMA 1988]. Some models of reliability oriented task allocation in redundant distributed computer systems [SHAT 1989. SHAT 1992] and estimating the reliability and availability of programs have also appeared in the literature. A 1- step algorithm GEAR (Generalized Evaluation Algorithm for Reliability), capable of computing several reliability parameters of a CS such a terminal-pair reliability, computer network reliability, distributed program reliability, and distributed system reliability, is also studied [KUMA 1993]. Software reliability allocation, based on several factors such as structure, utility, price, and cost, is proposed by Zahedi [ZAHE 1991]. Strategies are devised that achieve a maximal combination of safety and reliability [RAI 1982]. Efficient methods to optimize the system reliability have been reported in [PAIN 1995, RAGH 1985]. Reliability analysis for a distributed program is mentioned in [KUMA 1996, KE 1997]. A genetic algorithm based reliability optimization model for computer networks is proposed by Kumar et al [KUMA 1995a]. Further, an investigation of the problem of distributed- program reliability in various classes of distributed computing systems has been presented [LIN 1990]. Srinivasan and Jha suggested a new clustering-based safety and reliability driven heuristic for a heterogeneous distributed system [SRIN 1999].

CHAPTER-3

PROBLEMS

3.1 MOTIVATION

Communication occurs between entities (such as, file transfer packages, browsers, electronic mail software, etc.) in different systems. An entity is anything capable of sending or receiving information. A system is a physical object that contains one or more entities, such as computers, terminals, processors etc. Communication systems have homogenous and/or heterogeneous processors that are connected through a communication links. It provides the capability for the utilization of remote computing resources and allow for increased level of flexibility, reliability, and modularity. The distributed processing environments in which services provided for the network reside at multiple sites. Instead of single large machine being responsible for all aspects of process, each separate processor handles subset. In the distributed environments the programs or tasks are also often developed with the subsets of independent units under various environments. It has drawn tremendous attention in developing cost-effective and reliable applications to meet the desired requirements. Task allocation is the process of partitioning a set of programming modules into a number of processing groups, known as tasks, where each group executes on a separate processor. The general allocation problem is NP-complete. In these problems, suppose in any communication system there are 'm' tasks that are to be processed and it has at least 'n' processors which can process any of the m tasks but possibly in various time periods. Tasks are to be assigned to such processors so as to minimize the overall time to complete the execution of all 'm' tasks. Hungarian

approach [GILL 2002] is suggested for solving the assignment problem, but this approach is to be applicable only for m-tasks to the m-processors i.e. for balanced assignment problem. For the unbalanced assignment problem where m > n or n > m, the Hungarian method suggest to add the dummy tasks or processors to make the effectiveness matrix square.

The best-known research problem for such systems is the assignment problem, in which the total system time is to be minimized. The problem finding an assignment of tasks to nodes that minimizes total execution and communication cost (or time) was elegantly analyzed using a network flow model and network flow algorithms by [STON 1977, STON 1978] and a number of other researchers [YADA 2002, YADA 2003, KUMA 2001, SING 1999, SRIN 1999, PENG 1997, KUMA 1995a, KUMA 1995b, KUMA 1995c, KUMA 1995d, SAGA 1991, ZAHE 1991, BACA 1989, CHU 1969, KUMA 1999, KUMA 1996, KUMA 2002, KUMA 2006, CASA 1988, BOKH 1979] through various different algorithms. The load-balancing algorithms or load-leveling algorithms are based on the intuition that, for better resource utilization, it is desirable for the load in a distributed communication system to be balanced evenly. Thus a load-balancing algorithm tries to balance the total system load by transparently transferring the work load from heavily loaded nodes to lightly loaded nodes in an attempt to ensure good overall performance relative to some specific metric of system performance. When considering performance from the user point of view, the metric involved is often the response time of the processors. However, when performance is considered from the resource point of view, the metric involved is the total system throughput [SINH 2004]. Kumar et al [KUMA 1995a] suggested a modified and efficient task allocation approach in which authors have optimized the overall system

cost that include the execution and communication cost. Kumar et al [KUMA 1996] and Kumar [KUMA 1999] have suggested task allocation technique, which are dynamic in nature for optimizing the cost and reliability respectively. S. Srinivasan [SRIN 1999] stated that software or a program to be run on the distributed computing system is called a task, which is composed of intercommunication tasks, which allocated to the different processor in the system. Several other methods have been reported in the literature, such as, Integer Programming [DESS 1980], Branch and Bound technique [RICH 1982], Matrix Reduction technique [SAGA 1991], Reliability Optimization [LIN 2002, LYU 2002] Modeling [BIER 2002, FITZ 2002]. The Series Parallel Redundancy-Allocation problem has been studied with different approaches, such as, Non-Linear techniques [TILL 1977], and Heuristic techniques [COIT 1996, COIT 1998a, COIT 1998b, LO 1988].

Therefore it is recorded that the research problem for such systems is of the type tasks allocation problems, in which either system reliability is to be maximized or total system time is to be minimized or overall effectiveness of the computing systems is to be optimized under the pre specified constrains. Sager et al [SAGA 1991] discussed the tasks allocation problem in 1991; they considered a set of tasks and a set of processors, which are less in comparison to the number of tasks. They have suggested a heuristics approach by using matrix reduction technique for the allocation of tasks to the various processors, and obtained optimal execution cost. Kumar et al [KUMA 1995c, KUMA 1996] has studied the task allocation problem for a set of heterogeneous processors on which the set of tasks to be allocated in such a way that all the tasks got executed and load on each processor also maintained. These models gave the total assignment cost and results were much better as compared to

Sagar et al [SAGA 1991]. The complexity of the algorithm is also less as they have mentioned the complexity comparisons to other approaches in their papers. Kumar [KUMA 2001] and Singh et al [SING 1999] also extended the work on tasks allocation problem for analyzing the various performance parameters. Recently, Yuan-Shun and Levitin [YUAN 2007] has used optimal resource allocation for maximizing performance and reliability in tree structured grid series. The genetic algorithm is used for reliability-oriented task assignment by Chin-Ching Chiu et al [CHIN 2006]. The work related to redundancy allocation problem with a mix of components is studied by Onishi et al [ONIS 2007].

PROBLEM-I

OPTIMAL TIME EVALUATION OF CS: AN EXHAUSTIVE SEARCH APPROACH

3.1.1 Objective

Let the given Communication System (CS) consists of a set P = {p_1, p_2, p_3 p_4, ...,p_n } of n processors, interconnected by communication links and a set T = { t_1, t_2, t_3, t_4,.....,t_m } of m tasks. The processing Execution Time [ET] of individual tasks corresponding to each processor are given in the form of matrix ETM (,) of order m x n and the Communication Time [CT] is taken in the square symmetric matrices CTM (,) of order n respectively. The functions to measure ET and CT are then formulated. A procedure to assign all the tasks to the processors of the communicating systems based on execution time is to be designed in such a way that the overall time is to be optimizing under the pre specified constraints. The load on each processor has been also taken care off in order to balance the load of each processor.

3.1.2 Technique

Let the given system consists of a set P = {p_1, p_2, p_3 p_4, ...,p_n } of n processors, interconnected by communication links and a set T = { t_1, t_2, t_3, t_4,.....,t_m } of m tasks. The processing execution time of individual tasks corresponding to each processor are given in the matrix ETM (,) of order m x n. The communication time is taken in the square symmetric matrix CTM (,) of order n. Initially, we obtain the task combinations in order to make the set of task(s) equals to number of processor. These combinations shall be n * $^{m}C_{m-n}$ (= nl, say) and to be store in TCOMB (). Then we obtained an index, which is based on the processing time of the tasks to the processors for the execution of tasks to various processors, and also time of communication amongst the task to each of the combination, the maximum value of the index shall give us the optimal result. The assignment of tasks to processors may be done in different ways. To allocate the task to one of the processors, the minimum value of each row and column of ETM (,) is obtained. Let min {r_i} represent the minimum row time value corresponding to the tasks t_i and min {c_j} represent the minimum column time value for processor p_j. These values are then replaced to 0 in ETM (,). For allocation purpose a modified version of row and column assignment method, namely Kumar et al [KUMA 1995c] is employed which allocates a task to a processor where it has minimum execution time. The overall execution time [Etime] is expressed as the sum of execution and communication time of all the tasks as follows:

$$Etime = \left[\sum_{i=1}^{n} \left\{ \sum_{j=1}^{n} ET_{ij} x_{ij} \right\} + \sum_{j=1}^{n} \left\{ \sum_{j=1}^{n} CT_{ij} y_{ij} \right\} \right]$$

where,

$$ET = \left[\sum_{i=1}^{n} \left\{ \sum_{j=1}^{n} ET_{ij} x_{ij} \right\} \right]$$

where, $x_{ij} = \begin{cases} 1, \text{if } i^{th} \textit{ task is assigned to } j^{th} \text{ processor} \\ 0, \text{otherwise} \end{cases}$, and

$$CT = \left[\sum_{j=1}^{n} \left\{ \sum_{j=1}^{n} CT_{ij} y_{ij} \right\} \right]$$

$y_{ij} = \begin{cases} 1, \text{if the } \textit{task assigned} \text{ to processor i communicate with the task assigned to processor j} \\ 0, \text{otherwise} \end{cases}$

Index = $(\text{Etime})^{-1}$

3.1.3 Computational Algorithm

To given an algorithmic representation to the technique mentioned in the previous section, let us consider a system in which a set of m tasks $T = \{t_1, t_2, t_3, \ldots, t_m\}$ is to be executed on a set of n available processors $P = \{p_1, p_2, p_3, \ldots, p_n\}$.

Step-1:

Input: m, n, ETM (,), CTM (,), ; nar : = 0; T_{ass}: = { } ;

Step-2:

Select the task to form the combinations with other task(s), (say (t_i, t_k) & (t_k, t_i). Store it in TCOMB(l), where l=1(1) nl

Step-3:

for l:= 1 to nl do

begin

TCOMB(l)

Step-3.1:

for k : = 1 to n do

for i : = 1 to n do

begin

Add k^{th} row of ETM (,) to its i^{th} row. If all the values are become infinite, get next value of TCOMB (l) then repeat else Step-3.2.

end.

Step-3.2:

Modify CTM (,) and, ETM (,) as follow;

Step-3.2.1:

for k : = 1 to n do

for j : = 1 to n do

begin

Replace the corresponding values of k^{th} row and k^{th} column by zero in CTM (,) and then add k^{th} row to i^{th} row and k^{th} column to i^{th} column, after that delete the k^{th} row and k^{th} column from CTM (,).

Step-3.2.2:

Modify the ETM (,) by adding k^{th} row to i^{th} row and thereby deleting k^{th} row.

Step-3.2.3:

Store modified, ETM (,) and, CTM (,) to NETM (,) and NCTM(,) respectively.

end.

Step-4:

for k : = 1 to n do
for j : = 1 to n do
begin
Find the minimum of k^{th} row (Say mr_{kj}) of NETM (,) falling in j^{th} column and replace it by zero.
end.

Step-5:

for k : = 1 to n do
for j : = 1 to n do
begin
Find the minimum of j^{th} column (say mc_{kj}) of NETM (,), which lies in k^{th} row, and replace it by zero.
end.

Step-6:

for j : = 1 to n do
begin
for k : = 1 to n do
begin
Search for a row in NETM (,), which has only one zero say, at the position (k,j) and assign task(s) corresponding to this position.
nar : = nar + 1;
far(k) : = j;
T_{ass} : $= T_{ass} \cup \{t_k\}$;
end;
end.

Step-7:

for j : = 1 to n do
begin
for k : = 1 to n do
begin
Search for a column which has only one zero entry say, at the position (k,j) and assign task(s) corresponding to this position.
nar : = nar + 1;
far(k) : = j;
T_{ass} : $= T_{ass} \cup \{t_k\}$;
end;
end.

Step-8:

if nar ≠ n

Pick-up an arbitrary zero entry say, at the position (k,j) and assign task(s) corresponding to this position.

nar : = nar + 1;far(k) : = j;

T_{ass} : $= T_{ass} \cup \{t_k\}$;

else

Check column(s) positions of zero(s) in unassigned row (s). Check the row(s) for any previous assignment in the corresponding column(s). Find the minimum of the entire elements for the remaining rows and replace it zero and then go to step-6.

Step-9:

Evaluate Execution Time as;

ET : = 0.0;

for i : = 1 to n do

ET : = ET + ET(i, far(i))

ETT(l):= ET

Step-10:

Evaluate Communication Time as;

CT : = 0.0;

for i = 1 to n do

CT : = CT + CT(i, far(i))

CTT(l):= CT

Step-11:

Execution Time [Etime] and Index are thus calculated as:

Step-11.1:

Etime(l): = ETT(l)+ CTT(l)

Step-11.2:

Index(l): = 1 /Etime(l)

Step-12:

If l < nl then go to step-3.

Step-13:

List: Index (l); Etime (l);

end.

Step-14:

Stop.

3.1.4 Implementation

Example-I

Consider a system consisting of a set T = $\{t_1, t_2, t_3, t_4\}$ of 4 tasks and a set P = $\{p_1, p_2, p_3\}$ of 3 processors,

Step-1: Input: 4, 3

$$ETM(,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 8 & 12 & 7 \\ t_2 & 9 & 8 & 11 \\ t_3 & 12 & 9 & 6 \\ t_4 & 10 & 11 & 12 \end{array};$$

$$CTM(,) = \begin{array}{c|cccc} & t_1 & t_2 & t_3 & t_4 \\ t_1 & 0 & 3 & 6 & 9 \\ t_2 & 3 & 0 & 4 & 5 \\ t_3 & 6 & 4 & 0 & 7 \\ t_4 & 9 & 5 & 7 & 0 \end{array}$$

Step-2:

$$TCOMB(,) = \begin{bmatrix} 3 & 4 & 2 & 4 & 2 & 3 & 1 & 4 & 1 & 3 & 1 & 2 \\ 4 & 3 & 4 & 2 & 3 & 2 & 4 & 1 & 3 & 1 & 2 & 1 \end{bmatrix}$$

Step-3:

$$TCOMB(1) = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Step-3.1-3.2:

$$ETM(,) = \begin{array}{c|ccc} & p1 & p2 & p3 \\ t3 * t4 & 22 & 20 & 18 \end{array}$$

$$NCTM(,) = \begin{array}{c|ccc} & p1 & p2 & p3 \\ t1 & 0 & 3 & 6 \\ t2 & 3 & 0 & 4 \\ t3 * t4 & 6 & 4 & 0 \end{array}$$

$$NETM\ (,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 8 & 12 & 7 \\ t_2 & 9 & 8 & 11 \\ t_3 * t_4 & 22 & 20 & 18 \end{array}$$

Step-4 & 5:

On applying modified Hungarian method devised by Kumar et al [KUMA 1995c] to assign the task, min $\{r_i\}$ from NETM (,) for every i, $r_{13} = 7$, $r_{22} = 8$, $r_{33} = 18$. Making $r_{13} = r_{22} = r_{33} = 0$. Again min $\{c_j\}$ from NETM (,) for every j, are $c_{11} = 8$, $c_{22} = 0$, $c_{33} = 0$. Making $c_{11} = 0$, so that, we get,

$$NETM\ (,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 0 & 12 & 0 \\ t_2 & 9 & 0 & 11 \\ t_3 * t_4 & 22 & 20 & 0 \end{array}$$

Step-6, 7& 8:

After implementing assignment process, the first set of the allocation is thus obtained.

Tasks	→	Processors	ET	CT
t_1	→	p_1	08	03
t_2	→	p_2	08	06
$t_3 * t_4$	→	p_3	18	04

Step-9:

ETT (1) := 34

Step-10:

CTT (1) := 13

Step-11:

Etime (1) := 47

Step-12:

Index (1) := 0.02127650

Step-13:

On repeating the above process, the assignments and their corresponding related values of ET, CT, Etime, and Indexes are thus obtained, which are shown in the following table-1:

Table-1: Assignment Table

S. No.	ET	CT	Etime	INDEX	ASSIGNMENT-I		ASSIGNMENT-II		ASSIGNMENT-III	
1	34	13	47	0.02127650	t_1	p_1	t_2	p_2	$t_3 * t_4$	p_3
2	34	17	51	0.01960780	t_1	p_1	t_2	p_2	$t_4 * t_3$	p_3
3	33	13	46	0.02173910	t_1	p_1	t_3	p_3	$t_2 * t_4$	p_2
4	33	22	55	0.01818181	t_1	p_1	t_3	p_3	$t_4 * t_2$	p_2
5	34	17	51	0.01960780	t_1	p_3	t_4	p_1	$t_2 * t_3$	p_2
6	34	22	56	0.01785710	t_1	p_3	t_4	p_1	$t_3 * t_2$	p_2
7	32	13	45	0.02222222	t_2	p_2	t_3	p_3	$t_1 * t_4$	p_1
8	32	16	48	0.02083333	t_2	p_2	t_3	p_3	$t_4 * t_1$	p_1
9	31	17	48	0.02083333	t_2	p_2	t_4	p_1	$t_1 * t_3$	p_3
10	31	16	47	0.02127650	t_2	p_2	t_4	p_1	$t_3 * t_1$	p_3
11	34	16	50	0.02000000	t_3	p_3	t_4	p_2	$t_2 * t_1$	p_1
12	34	22	56	0.01785710	t_3	p_3	t_4	p_2	$t_1 * t_2$	p_1

Step-14:

Stop.

Example-II

The results of a system which consist a set T = {t_1, t_2, t_3, t_4, t_5} of 5 tasks and a set P = {p_1, p_2, p_3}, of 3 processors where,

$$ETM\ (,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 8 & 12 & 7 \\ t_2 & 9 & 8 & 11 \\ t_3 & 12 & 9 & 6 \\ t_4 & 10 & 11 & 12 \\ t_5 & 7 & 6 & 2 \end{array}$$

$$CTM\ (,) = \begin{array}{c|ccccc} & t_1 & t_2 & t_3 & t_4 & t_5 \\ t_1 & 0 & 3 & 6 & 9 & 8 \\ t_2 & 3 & 0 & 4 & 5 & 6 \\ t_3 & 6 & 4 & 0 & 7 & 9 \\ t_4 & 9 & 5 & 7 & 0 & 5 \\ t_5 & 8 & 6 & 9 & 5 & 0 \end{array}$$

The following tables shows the results as obtain after implementing the present algorithm

Tasks	→	Pr *ocessors*	*ET*	*CT*
$t_1 * t_4$	→	p_1	18	3
t_2	→	p_2	8	6
$t_3 * t_5$	→	p_3	8	4

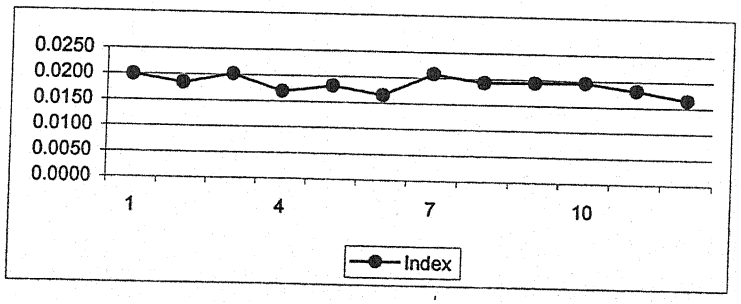
Tasks	→	Pr*ocessors*	*Etime*	*Index*
$t_1 * t_4$	→	p_1		
t_2	→	p_2	47	0.021276595
$t_3 * t_5$	→	p_3		

3.1.5 Conclusion

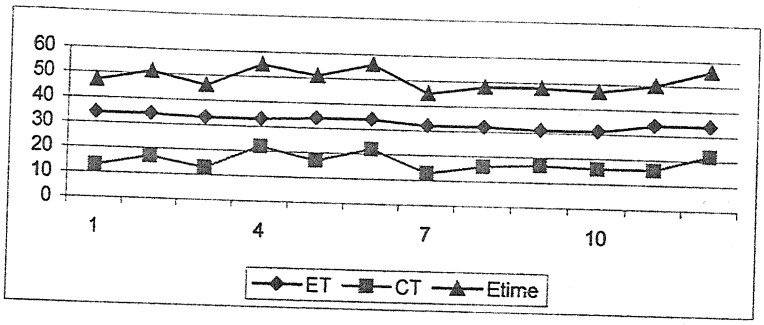
The first problem of the thesis discusses an assignment model through optimization technique for the performance enhancement of communication system. In this problem we have chosen such a communication system in which numbers of tasks are more than the number of processors. The present method deals the case when the index is based on the processing time of the tasks to the processors for the execution of tasks to various processors and also communication time amongst the tasks. The method is presented in computational algorithmic form and implemented on the several sets of input data to test the performance and effectiveness of the algorithm. The optimal result of the Example-I are shown in the following table:

Tasks	→	Pr*ocessors*	*Etime*	*Index*
t_3	→	p_3		
t_2	→	p_2	45	0.02222222
$t_1 * t_4$	→	p_1		

The graphical representation of the results mentioned in Table-1 of Step-12 of the implementation part of this paper is shown in the following graphs:



Graph-1: Index Graph



Graph-2: Execution Time Graph

The following table shows the results of Example-II given in its implementation part of problem-I of this chapter, as obtain after implementing the present algorithm:

Tasks	→	Pr*ocessors*	*Etime*	*Index*
$t_1 * t_4$	→	p_1		
t_2	→	p_2	47	0.0212765
$t_3 * t_5$	→	p_3		

3.1.5.1 Comparison

The run time complexity of the algorithm is measured O ($n * {}^{m}C_{m-n}$) time. Our time complexity is better than O (m^2 n) of [SAGA 1991] and O (n^m) to [PENG 1997, RICH 1982]. The performance of the algorithm is compared with that of [SAGA 1991] and [PENG 1997, RICH 1982] and results for complexity comparison are shown below:

Taks	O ($n * {}^{m}C_{m-n}$)	O(m^2 n)	O (n^m)
(m,n)	Pr*esent* – *A*lg	[*SAGA*1991]	[*PENG*1997, *RICH*1982]
(5,3)	30	75	243
(6,3)	60	108	729
(7,3)	105	147	2187
(8,3)	168	192	6561
(10,4)	630	400	1048576
(10,5)	756	500	09765625

PROBLEM-II

OPTIMIZING EXECUTION TIME OF CS: MATRIX PARTITIONING APPROACH

3.2.1 Objective

Consider a communication system which consist a set $P = \{p_1, p_2, \ldots p_n\}$ of "n" processors, interconnected by an arbitrary network. The processors have local memory only and do not share any global memory. The processor graph is a convenient abstraction of the processors together with interconnection network. It has processors as nodes and there is a weighted edge and distance between two nodes if the corresponding processors can communicate with each other. The weight w_{ij} and distance d_{ij} on the edge between processors p_i and p_j represent the delay time involved in sending or receiving the message of d_{ij} length from one processor to another. In order to have an approximate estimate of this time, irrespective of the two processors, we use the average of the weights on all the edges in the processor graph.

A set $T = \{t_1, t_2, \ldots t_m\}$ of "m" tasks is considered at hand to be executed on "n" processors. The Execution Time (ET) of these tasks on all the processors is given in the form of Execution Time Matrix [ETM (,)] of order m x n. The Communication Time (CT) is taken in the form of a symmetric matrix named as Communication Time Matrix [CTM (,)], which is of order m. In order to make the best use of the resources of the system for that we would like to distribute the load on each processor in such a way that allocated load on the processors should be evenly balanced. The proposed model discusses the following issues:

- Developing the method to form the sub problems,
- Formulating the method to assign all tasks,

- Formulating the Time function to measure ET,
- Formulating the Time function to measure CT.

3.2.2 Technique

Since the numbers of task are more than the number of processors, so that we divide the problem of unbalanced tasks assignment in to sub problems, which becomes the balanced tasks assignment problems. Obtain the sum of each row and each column except infinity time (infinity time should be kept aside) form the ETM (,) and store the results in to Sum_Row() and Sum_Column(), each of them one dimensional arrays. Select the first set of tasks, (this set of tasks shall contain only as many tasks as the number of processor in the communication system) on the basis of minimum time against the tasks in the Sum_Row() array. Store the result in to ETM (, ,) a two dimensional array. Repeat the process until remaining tasks are either less than or equal to the number of processors. If the tasks are equal to the processors of the communication system then it will becomes the last sub problem, else to form the last problem we have to delete the column (processor) form ETM(,) on the basis of Sum_Column() array. Assignment of these tasks for each sub problem, we apply the Kumar et al algorithm [KUMA 1995c]. The communication time of those tasks, which are allocated on the same processor, becomes zero. For each sub problem we calculate the exaction time and communication time of each processor and store the result in a linear array PET(j) and PCT(j) respectively where j= 1,2,...n. Finally, sum up the value of PET (j) and PCT (j), (j=1,....,n) to obtain Etime. It is the total optimal time for the complete assignments.

$$PET(j) = \sum_{i=1}^{n} \sum_{j=1}^{n} ET_{ij} \, x_{ij} \; ;$$

$$PCT(j) = \sum_{i=1}^{n} \sum_{j=1}^{n} CT_{ij}\ y_{ij}\ ;$$

$$Etime = \left[\sum_{i=1}^{n} \left\{ \sum_{j=1}^{n} ET_{ij}\, x_{ij} \right\} + \sum_{j=1}^{n} \left\{ \sum_{j=1}^{n} CT_{ij}\, y_{ij} \right\} \right]$$

where, $x_{ij} = \begin{cases} 1, \text{ if } i^{th} \textit{ task } \textit{is assigned} \text{ to } j^{th} \text{ processor} \\ 0, \text{ otherwise} \end{cases}$, and

$y_{ij} = \begin{cases} 1, \text{ if the } \textit{task } \textit{assigned} \text{ to processor i communicat e with the task assigned to processor j} \\ 0, \text{ otherwise} \end{cases}$

3.2.3 Computational Algorithm

To given an algorithmic representation to the proposed method as discussed, we have to concentrate on such a system which consist a set P = {p_1, p_2, ……., p_n} of n processor and a set T = {t_1, t_2,……,t_m} of m executable tasks which are to be processed by any one of the processor of the system.

Step-1:

Input: m, n. ETM (,), CTM(,)

Step-2:

Obtain the sum of each row of the ETM(,) in such a way that, if any time values (s) is (are) ∞ then keeping it aside along with sum of that row (just to avoid the condition sum ET + ∞ = ∞. Store the results in one-dimensional array Sum _Row (,) of order m.

Step-3:

Obtain the sum of each column of the ETM (,), in such a way that if any time value (s) is (are) ∞ then keeping it aside along with product of that column (just to avoid the condition ET + ∞ = ∞). Store the results in one-dimensional array Sum _Column (,) of order n.

Step-4:

Partitioned the execution time matrix ETM (,) of order m x n in to sub matrices such that the order of these matrices become square i.e. number of row should be equal to number of column. Partitioning to be made as mentioned in the following steps.

Step-4.1:

Select the n task on basis of Sum _Row (,) array i.e. select the 'n' task corresponding to most minimum sum to next minimum sum, if there is a tie select arbitrarily. (For the cases in which product is ET + ∞ = ∞, minimum value depends only ET and the impact of ∞ is to be neglected.

Step-4.2:

Store the result in the two dimensional array ETM (, ,) to form the sub matrices of the sub problems.

Step-4.3:

If all the tasks are selected then go to step 4.7 else steps 4.4

Step-4.4:

Repeat the step 4.1 to 4.3 until the number of task become less than n.

Step-4.5:

Select the remaining task say r, $r < n$, select the r processors on the basis of Sum_ Column (,) array i.e. the processors corresponding to the most minimum sum to next minimum, if there is a tie select arbitrarily (for the cases in which product is $ET + \infty = \infty$, minimum value depend only ET and the impact of ∞ is to be neglected.

Step-4.6:

Store the result in the two dimensional array ETM (,,), which is a last sub problems.

Step-4.7:

List of all the sub problems formed through Step 4.1 to 4.6 and repeat step 5 to step 12 to solve each of these sub problems.

Step-5:

Find the minimum of each row of ETM (, ,) and replace it by 0.

Step-6:

Find the minimum of each column of ETM (, ,) and replace it by 0.

Step-7:

Search for a row in ETM (, ,) which has only one zero and assign the task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position.

Step-8:

Search for a column in ETM (, ,) which has only one zero and assign the task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position.

Step-9:

Check whether nar = n if not than pickup an arbitrary 0 and assigned task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position, else, Check column (s) position of 0's in unassigned row(s). Check the row(s) for any previous assignment in the corresponding column(s). Find the minimum of the entire elements for the remaining rows and replace it zero, Go to Step-7, Else Step-10.

Step-10:

Evaluate Execution Time [PET ( )].

Step-11:

Evaluate Communication Time [PCTC ( )].

Step-12:

Execution Time (Etime) are thus calculated as:

Etime = PET ( ) + PCT ( )

Step-13:

Stop.

3.2.4 Example

Consider a communication system which is consisting of a set P = {p_1, p_2, p_3} of "n = 3" processors connected by an arbitrary network. The processors only have local memory and do not share any global memory. The processor connections graph is a depicted in figure-1 and tasks execution graph also pictorially depicted in figure-2. A set T = {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8,} of "m = 8" executable tasks which may be portion of an executable code or a data file. The communication graph is depicted in figure-3.

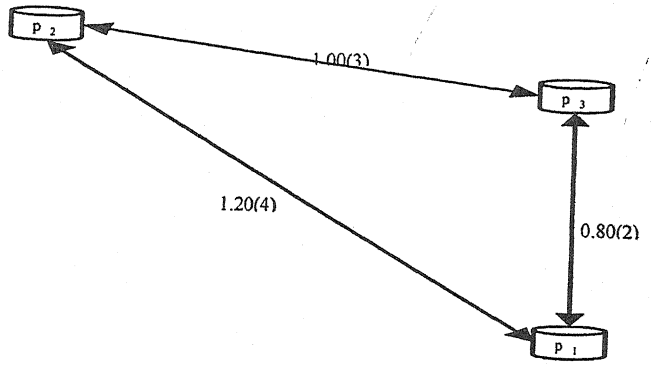


Figure- 1: Processors Graphs

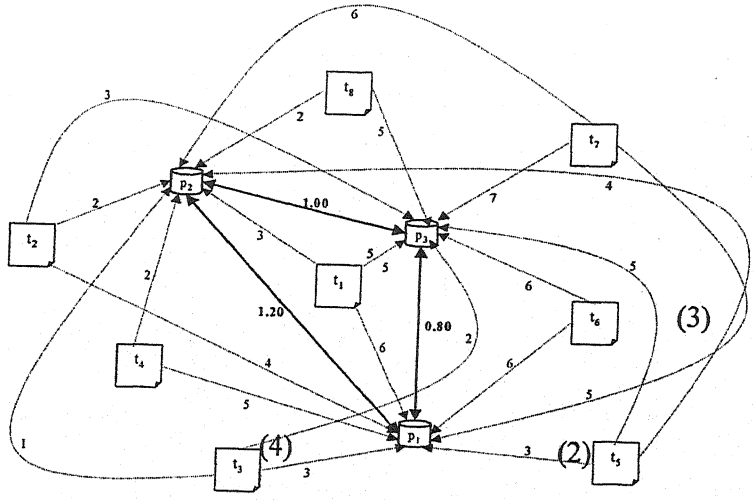


Figure – 2: Tasks Execution Graph

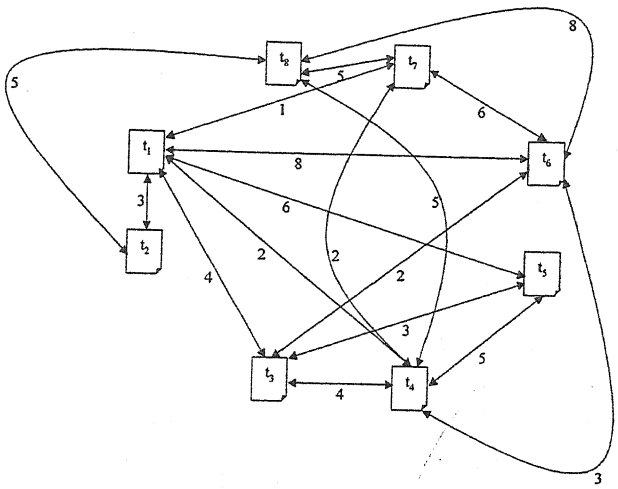


Figure – 3: Inter Tasks Communications Graphs

Step-1

On converting its matrix representations, we have, m = 8, n = 3,

ETM(,) =

	p_1	p_2	p_3
t_1	6	3	5
t_2	4	2	3
t_3	3	1	2
t_4	5	2	∞
t_5	3	4	2
t_6	6	∞	6
t_7	5	6	7
t_8	∞	2	5

CTM(,)=

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1	0	3	4	2	6	8	1	0
t_2	3	0	0	0	0	0	0	5
t_3	4	0	0	4	3	2	0	0
t_4	2	0	4	0	5	3	2	5
t_5	6	0	3	5	0	0	0	0
t_6	8	0	2	3	0	0	6	8
t_7	1	0	0	2	0	6	0	5
t_8	0	5	0	5	0	8	5	0

Step-2

Obtain the sum of each row and column of ETM (,), as:

$$Sum\_Row = \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 \\ 14 & 9 & 6 & 7+\infty & 9 & 12+\infty & 18 & 7+\infty \end{matrix}$$

$$Sum\_Column = \begin{matrix} p_1 & p_2 & p_3 \\ 32+\infty & 20+\infty & 30+\infty \end{matrix}$$

Step-3

We partitioned the matrix ETM (,) to define the first sub problem ETM (1,) by selecting rows corresponding t_3, t_4, t_8 and second sub problem ETM (2,) by selecting rows corresponding t_2, t_5, t_6 and on the basis of the Sum_Column, by deleting columns corresponding p_1, and after the selecting the remaining two tasks t_1, t_7 to form the last sub problem ETM (3,), as there were only two tasks, for which we required two processors. So that the modified matrices are as;

Sub Problem-I:

ETM (1 ,) =		p_1	p_2	p_3
	t_3	3	1	2
	t_4	5	2	∞
	t_8	∞	2	5

Sub Problem-II:

ETM (2 ,) =		p_1	p_2	p_3
	t_2	4	2	3
	t_5	3	4	2
	t_6	6	∞	6

and, Sub Problem-III:

ETM 3(,) =		p_2	p_3
	t_1	3	5
	t_7	6	7

Step-4 & 5:

On applying modified Hungarian method devised by Kumar et al [KUMA 1995c] to assign the tasks, on the basis of min $\{r_i\}$ and min $\{c_j\}$ from execution time matrices for every i and j. We put $r_{ij} = 0$ and $c_{ij} = 0$, for every i and j. So that, the modified matrices for each sub problem are mentioned below:

Sub Problem-I:

NETM (1,) =		p_1	p_2	p_3
	t_3	0	0	0
	t_4	5	0	∞
	t_8	∞	0	5

Sub Problem-II:

NETM (2,) =		p_1	p_2	p_3
	t_2	4	0	3
	t_5	3	4	0
	t_6	0	∞	6

and, Sub Problem-III:

NETM (3,) =		p_2	p_3
	t_1	0	0
	t_7	0	7

Step-6, 7& 8:

After implementing assignment process, the solution set, of the each of sub problem, the allocation is thus obtained.

Solution for the Sub Problem-I:

Tasks →	Processors	ET
t_3 →	p_3	02
t_4 →	p_1	05
t_8 →	p_2	02

Solution for the Sub Problem-II:

Tasks →	Processors	ET
t_2 →	p_2	02
t_5 →	p_3	02
t_6 →	p_1	06

Solution for the Sub Problem-III:

Tasks →	Processors	ET
t_1 →	p_3	05
t_7 →	p_2	06

Step-9:

After implementing the process, we obtain the following set of complete assignments along with execution and communication times of each processor.

Processor→ Tasks	PET()	PCT ()
$p_1 \rightarrow t_4; t_6$	11	43
$p_2 \rightarrow t_2; t_7; t_8$	10	26
$p_3 \rightarrow t_1; t_3; t_5$	9	25

Step-10:

PET ( ) := 30
PCT ( ) := 94
Etime := 124

Step-12:

Stop.

3.2.5 Conclusion

The model discussed in this problem provide an optimal solution for assigning a set of "m" tasks of a program to a set of "n" processors where m > n in a communication system that to minimize the overall time of the system and the load of all allocated tasks on all the processors evenly balanced. The communication time and execution time on different processors has been obtained.

Processors of communication system	p_1	p_2	p_3
Execution time of each processor PET(,)	11	10	09
Communication time of each processor PCT(,)	43	26	25
Total time of each processor time(,)	54	36	34

The model addressed in this problem is based on the consideration of execution and communication times of the tasks to the processors. Keeping in view we suggested a modified method to assign all the tasks as per the required availability of processors so that none of the tasks get remains unexecuted and the present approach does not require to adding dummy processors. The algorithm is capable to improve the performance of the given communication system by optimally assigning the tasks to various processors of the system. The final results of the problem considered in the implementation of the algorithm are mentioned as,

Tasks	→	Pr*ocessors*	*ET*	*CT*	*Ttime*	*Etime*
$t_4;t_6$	→	p_1	11	43	54	
$t_2;t_7;t_8$	→	p_2	10	26	36	124
$t_1;t_3;t_5$	→	p_3	09	25	34	

The model discussed here, would be useful to the network system designer working in the field of distributed communication systems. The developed model is programmed in C++ and implemented the several sets of input data to test the effectiveness and efficiency of the algorithm. It is recorded that the model is suitable for arbitrary number of processors with the random program structure.

3.2.5.1 Comparison

The run time complexity of the algorithm is measured O ($6mn-n^2$) time. Here it is concluded that it remains same as [KUMA 1998, YADA 2002]. Our time complexity is better than O ($m^2 n$) of [SAGA 1991] and O (n^m) to [PENG 1997, RICH 1982]. The performance of the algorithm is compared with that of [SAGA 1991] and [PENG 1997, RICH 1982] and result in both cases, i.e. when tasks increases against the fixed processors and the case in which the tasks remain fix, while increase in processors are shown below:

Case-1: For fixed processors n = 3

Taks	$O(6mn - n^2)$	$O(m^2 n)$	$O(n^m)$
m	Pr*esent – A*lg	[*SAGA*1991]	[*PENG*1997, *RICH*1982]
5	81	75	243
6	99	108	729
7	117	147	2187
8	135	192	6561
9	153	243	19683
10	171	300	59049

Case-2: For fixed tasks m = 10

Pr*ocessors*	$O(6mn - n^2)$	$O(m^2 n)$	$O(n^m)$
n	Pr*esent – A*lg	[*SAGA*1991]	[*PENG*1997, *RICH*1982]
3	171	300	59049
4	224	400	1048576
5	275	500	9765625
6	324	600	60466176
7	371	700	282475249
8	416	800	1073741824

PROBLEM-III

A NEW APPROACH FOR ASSIGNMENT IN CS: BASED ON TASKS WEIGHT

3.3.1 Objective

Consider a communication system which consist a set $P = \{p_1, p_2, \ldots p_n\}$ of "n" processors, interconnected by an arbitrary network. A set $T = \{t_1, t_2, \ldots t_m\}$ of "m" tasks is considered at hand to be executed on "n" processors. The Task Execution Time [TET] of these tasks on all the processors is given in the form of Task Execution Time Matrix [TETM (,)] of order m x n. The Task Weight (WT) are taken in the form of a linear array named as Task Weight Matrix [TWM (,)], which is of order m. In

order to make the best use of the resources of these systems for that we would like to distribute the load on each processor in such a way that allocated load on the processors should be evenly balanced.

3.3.2 Proposed Method

Since the number of task are more than the number of processors, so that we divide the problem of unbalanced tasks assignment in to sub problems, which are to be balanced tasks assignment problems. First of all arrange the weights of the tasks in descending order and store the result in a linear array namely, wt_seq(). Select the first set of tasks, (this set of tasks shall contain only as many tasks as the number of processor in the communication system) on the basis of the wt_seq () array. Store the result in to TETM (, ,) a two dimensional array. Repeat the process until remaining tasks are either less than or equal to the number of processors. If the tasks are equal to the processors of the communication system then it will becomes the last sub problem, else to form the last problem we have to add dummy task (s) to make it square matrix. Assignment of these tasks for each sub problem, we apply the Kumar et al algorithm [KUMA 1995c]. For each sub problem we calculate the task exaction time and store the result in a linear array TET (j) where j= 1, 2,.....n. Finally, obtain the Etime, which is the product of TET (j) and WT (j), (j=1,....,n). It is the total optimal time for the complete assignments.

$$TET(j) = \sum_{i=1}^{n} \sum_{j=1}^{n} TET_{ij} \, x_{ij} \; ;$$

$$Etime = \left[\sum_{i=1}^{n} \left\{ \sum_{j=1}^{n} WT_{ij} * TET_{ij} x_{ij} \right\} \right]$$

where, $x_{ij} = \begin{cases} 1, \text{ if } i^{th} \text{ task is assigned to } j^{th} \text{ processor} \\ 0, \text{ otherwise} \end{cases}$

3.3.3 Algorithm

To given an algorithmic representation to the proposed method as discussed above of this chapter, we consider a system which consist a set P = {p_1, p_2, ..., p_n}of n processor and a set T = {t_1, t_2,.....,t_m}of m executable tasks which are to be processed by any one of the processor of the system.

Step-1:

Input: m, n. TETM (,), TWM(,)

Step-2:

Arrange the weight of the task in descending order and store the result in wt_seq () array.

Step-3:

Partitioned the execution time matrix TETM (,) of order m x n to sub matrices such that the order of these matrices become square i.e. number of row should be equal to number of column. Partitioning to be made as mentioned in the following steps.

Step-4.1:

Select the n task on basis of wt _seq (,) array i.e. select the 'n' task corresponding to most minimum weight to next minimum , if there is a tie select arbitrarily.

Step-4.2:

Store the result in the two dimensional array ETM (, ,) to form the matrices for the sub problems.

Step-4.3:

If all the tasks are selected then go to step 4.7 else steps 4.4

Step-4.4:

Repeat the step 4.1 to 4.3 until the number of task become less than n.

Step-4.5:

Add dummy task in order the make last sub problem as balanced problem.

Step-4.6:

Store the result in the two dimensional array TETM (,,), which is a last sub problem.

Step-4.7:

List of all the sub problems formed through Step 4.1 to 4.6 and repeat step 5 to step 11 to solve each of these sub problems.

Step-5:

Find the minimum of each row of TETM and replace it by 0.

Step-6:

Find the minimum of each column of TETM and replace it by 0.

Step-7:

Search for a row in TETM, which has only one zero and assign the task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position.

Step-8:

Search for a column in TETM, which has only one zero and assign the task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position.

Step-9:

Check whether nar = n if not than pickup an arbitrary 0 and assigned task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position, else, Check column (s) position of 0's in unassigned row(s). Check the row(s) for any previous assignment in the corresponding column(s). Find the minimum of the entire elements for the remaining rows and replace it zero, go to Step-7; else Step-10.

Step-10:

Evaluate Task Execution Time [TET ( )].

Step-11:

Execution Total Time [Etime] is thus calculated as:
Etime = ET * WT

Step-12:

Stop.

3.3.4 Implementation Of The Algorithm

Consider a communication system which is consisting a set P = {p_1, p_2, p_3} of "n = 3" processors connected by an arbitrary network. A set T = {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8,} of "m = 8" executable tasks which may be portion of an executable code or a data file. The tasks are different in nature and size so as the weights of tasks are defined. The weights of the tasks are given in Task Weight Matrix TWM (,) of order 1 x 8 The execution time per unit weight of the tasks on various processor is mentioned in Task Execution Time Matrix, namely, TETM (,) of order 8 x 3.

TETM(,) =

	p_1	p_2	p_3
t_1	6	3	5
t_2	4	2	3
t_3	3	1	2
t_4	5	2	∞
t_5	3	4	2
t_6	6	∞	6
t_7	5	6	7
t_8	∞	2	5

		weights
	t_1	70
	t_2	90
	t_3	30
TWM(,) =	t_4	20
	t_5	40
	t_6	50
	t_7	60
	t_8	20

Step-2

The weights of the tasks are arranged in descending order and store the result in wt_seq () linear array.

$wt\_seq=$	t_4	t_8	t_3	t_5	t_6	t_7	t_1	t_2
	20	20	30	40	50	60	70	90

Step-3

We partitioned the matrix TETM (,) to define the first sub problem TETM (1,) by selecting rows corresponding t_4, t_8, t_3 and second sub problem TETM (2,) by selecting rows corresponding t_5, t_6, t_7 and by the selecting the remaining two tasks t_1, t_2, t_d, to form the last sub problem TETM (3,), Where as t_d represent the dummy task that is required to be adding to the last sub problem TETM (3,), to make it balance problem. So that the modified matrices are as;

Sub Problem-I:

		p_1	p_2	p_3
	t_4	5	2	∞
TETM (1 ,) =	t_8	∞	2	5
	t_3	3	1	2

Sub Problem-II:

		p_1	p_2	p_3
	t_5	3	4	2
TETM (2 ,) =	t_6	6	∞	6
	t_7	5	6	7

and, Sub Problem-III:

TETM (3 ,) =		p_1	p_2	p_3
	t_1	6	3	5
	t_2	4	2	3
	t_d	0	0	0

Step-4 & 5:

On applying modified Hungarian method devised by Kumar et al [KUMA 1995c] to assign the tasks, on the basis of min $\{r_i\}$ and min $\{c_j\}$ from task execution time matrices for every i and j. We put $r_{ij} = 0$ and $c_{ij} = 0$, for every i and j. So that, the modified matrices for each sub problem are mentioned below:

Sub Problem-I:

TETM (1 ,) =		p_1	p_2	p_3
	t_4	0	0	∞
	t_8	∞	0	0
	t_3	0	5	0

Sub Problem-II:

TETM (2 ,) =		p_1	p_2	p_3
	t_5	3	0	0
	t_6	0	∞	0
	t_7	0	6	7

and,

Sub Problem-III:

TETM (3 ,) =		p_1	p_2	p_3
	t_1	3	0	2
	t_2	1	0	0
	t_d	0	3	0

Step-6, 7& 8:

After implementing assignment process, the solution set, of the each of sub problem, the allocation is thus obtained.

Tasks → Processors	ET	WT	Total Time
Solution for the Sub Problem-I:			
$t_4 \rightarrow p_1$	5	20	100
$t_8 \rightarrow p_2$	2	20	40
$t_3 \rightarrow p_3$	2	30	60
Average Time			100
Solution for the Sub Problem-II:			
$t_5 \rightarrow p_2$	4	40	160
$t_6 \rightarrow p_3$	6	50	300
$t_7 \rightarrow p_1$	5	60	300
Average Time			300
Solution for the Sub Problem-III:			
$t_1 \rightarrow p_2$	3	70	210
$t_2 \rightarrow p_3$	3	90	270
Average Time			270

Step-9:

After implementing the process, we obtain the following set of complete assignments along with execution and communication times of each processor.

Processor→ Tasks	PET()	Optimal Execution Time
$p_1 \rightarrow t_4; t_7$	400	630
$p_2 \rightarrow t_1; t_5; t_8$	410	
$p_3 \rightarrow t_2; t_3; t_6$	630	

Step-10 & 11:

Etime: = 630

Step-12:

Stop.

3.3.5 Conclusion

The model discussed here provide an optimal solution for assigning a set of "m" tasks of a program to a set of "n" processors where m > n in a communication system that to maximize the overall time of the system and the load of all allocated tasks on all the processors evenly balanced. The model addressed here is based on the

consideration of task execution time to the processors. The task weights are also defined that represent their nature, size, etc. Keeping in view we suggested modified method to improve the performance of communication system by assigning all the tasks as per the required availability of processors so that none of the tasks get remains unexecuted and the present approach does not require to adding dummy processors, however dummy task may be added if the last sub problem contains less task as compare to processor. The final results of the problem considered in the implementation of the algorithm are mentioned as,

Tasks	→	Pr*ocessors*	*Ttime*	*Etime*
$t_4;t_6$	→	p_1	400	
$t_1;t_5;t_8$	→	p_2	410	630
$t_2;t_3;t_6$	→	p_3	630	

The model discussed here, would be useful to the network system designer working in the field of communication systems and other related systems such as computing system, computer communication networks, distributed systems etc. The developed model is programmed in C++ and implemented the several sets of input data to test the effectiveness and efficiency of the algorithm. It is recorded that the model is suitable for arbitrary number of processors with the random program structure.

PROBLEM-IV

RELIABILITY IMPROVEMENT OF CS: MATRIX PARTITIONING APPROACH

3.4.1 Objective

The objective of this problem is to maximize the total execution reliability by allocating all the tasks optimally to the communication system. For that purpose we

considered execution unreliability and uses the method of matrix partitioning. The Execution Unreliability (EUR) is presented by an array in the form of Execution Unreliability Matrix EURM (,) of order m x n. A procedure is then formulated to assign the tasks to the processors of the communication systems based on execution unreliability and is to be designed so that the overall reliability becomes optimized. This problem presented a modified method to assign all the tasks as per the required availability of processors so that none of the tasks get remains unexecuted as number of tasks is more than number of processors.

3.4.2 Technique

For developing the technique we have form the sub problems using matrix partitioning i.e. each of the sub problem is of the type of balanced assignment problem. Devising a method to assign all tasks to the processors and formulating the unreliability function to measure EUR. Since the number of task is more than the number of processors, so that we divide the problem of unbalanced tasks assignment in to sub problems using matrix partitioning, this gives us balanced tasks assignment problems. First of all obtain the product of each row and each column except, the position where, the unreliability is zero (zero unreliability should be kept aside with the product of row or column) from the EURM(,) and store the results into Product_Row() and Prouct_Column(), each of them are one dimensional arrays. Select the first set of tasks, (this set of tasks shall contain only as many tasks as the number of processor in the communication system) on the basis of minimum unreliability against the tasks in the Product_Row() array. Store the result in to EURM (, ,) a two dimensional array. Repeat the process until remaining tasks are either less than or equal to the number of processors. If the tasks are equal to the

processors of the systems then it will becomes the last sub problem, else to form the last problem we have to delete the column (processor) from EURM(, ,) on the basis of Product_Column( ) array i.e. this set shall contain only as many processors as number of tasks left, so that we delete the processors that have maximum unreliability in the Product_Column() array. For allocation purpose a modified version of row and column assignment method devised by Kumar et al [KUMA 1995c] is employed which allocates a task to a processor where it has minimum execution unreliability. For each sub problem calculate the execution unreliability of each processor and store the result in a linear array PEUR(j) where j= 1,2,...n.

$$PEUR(j) = \prod_{i=1}^{n} \{ \sum_{j=1}^{n} eur_{ij} x_{ij} \} ;$$

where, $x_{ij} = \begin{cases} 1, \text{ if } i^{th} \textit{ task is assigned to } j^{th} \text{ processor} \\ 0, \text{ otherwise} \end{cases}$, and

3.4.3 Algorithm

Consider a communication system, which consist a set P = {p_1, p_2, ..., p_n}of n processor and a set T = {t_1, t_2,.....,t_m} of m executable tasks which are to be processed by any one processor of the system.

Step-1:
Input: m, n. ERM (,).

Step-2:
Obtain the product of each row of the EURM(,) in such a way that, if any unreliability(ies) is (are) zero then keeping it aside along with product amount of that row (just to avoid the condition EUR * 0 = 0.) Store the results in one-dimensional array Product _Row (,) of order m.

Step-3:
Obtain the product of each column of the EURM (,), in such a way that if any unreliability (ies) is (are) zero then keeping it aside along with

product of that column (just to avoid the condition EUR * 0 = 0). Store the results in one-dimensional array Product _Column (,) of order n.

Step-4:

Partitioned the execution unreliability matrix EURM (,) of order m x n to sub matrices such that the order of these matrices become square i.e. number of row should be equal to number of column. Partitioning to be made as mentioned in the following steps.

Step-4.1:

Select the n task on basis of Product _Row (,) array i.e. select the 'n' task corresponding to most minimum product to next minimum product, if there is a tie select arbitrarily. (For the cases in which product is EUR * 0, minimum value depends only EUR and the impact of zero is to be neglected).

Step-4.2:

Store the result in the two dimensional array EURM (, ,) to form the sub matrices of the sub problems.

Step-4.3:

If all the tasks are selected then go to step 4.7 else steps 4.4

Step-4.4:

Repeat the step 4.1 to 4.3 until the number of task become less than n.

Step-4.5:

Select the remaining task say r, r < n, select the r processors on the basis of Product_ Column (,) array i.e. the processors corresponding to the most maximum product to next maximum, if there is a tie select arbitrarily (for the cases in which product is EUR * 0, maximum value depend only EUR and the impact of zero is to be neglected).

Step-4.6:

Store the result in the two dimensional array EURM (,,), which is a last sub problems.

Step-4.7:

List of all the sub problems formed through Step 4.1 to 4.6 and repeat step 5 to step 11 to solve each of these sub problems.

Step-5:

Find the minimum of each row of EURM (, ,) and replace it by 0.

Step-6:

Find the minimum of each column of EURM(, ,) and replace it by 0.

Step-7:

Search for a row in EURM(, ,), which has only one zero and assign the task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position.

Step-8:

Search for a column in EURM(, ,), which has only one zero and assign the task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position.

Step-9:

Check whether nar = n if not than pickup an arbitrary 0 and assigned task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position, else, Check column (s)

position of 0's in unassigned row(s). Check the row(s) for any previous assignment in the corresponding column(s). Find the minimum of the entire elements for the remaining rows and replace it zero, go to Step-7; Else Step-10.

Step-10:

Evaluate Execution Unreliability [EUR]

Step-11:

Execution Reliability (Ereliability) is thus calculated as:

Ereliability = 1 - EUR

Step-12:

Stop.

3.4.4 Implementation

Example

Consider a system which consists a set $T = \{t_1, t_2, t_3, t_4, t_5\}$ of 5 tasks and a set $P = \{p_1, p_2, p_3\}$ of 3 processors, where,

Step-1: Input: 5, 3

$$EURM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.03 & 0.04 & 0.06 \\ 0.07 & 0.02 & 0.08 \\ 0.02 & 0.09 & 0.06 \\ 0.03 & 0.07 & 0.02 \\ 0.08 & 0.05 & 0.04 \end{array}$$

Step-2:

Obtain the product of each row and column of EURM (,), i .e. the products of each row and each column are as:

$$Product\_Row() = \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \\ 0.000072 & 0.000112 & 0.000108 & 0.000042 & 0.000160 \end{array}$$

$$Product\_Column() = \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.0000001008 & 0.0000002520 & 0.0000002304 \end{array}$$

Step-3:

We partitioned the matrix EURM (, ,) to define the first sub problem EURM (, ,) by selecting rows corresponding to t_2, t_3, t_5 and second sub problem EURM (, ,) by selecting rows corresponding to the tasks t_1, t_4 and by deleting columns corresponding to p_1. So that the modified matrices for each sub problems are as;

Sub Problem-I:

$$EURM\ (1,) = \begin{array}{c} \\ t_2 \\ t_3 \\ t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.07 & 0.02 & 0.08 \\ 0.02 & 0.09 & 0.06 \\ 0.08 & 0.05 & 0.04 \end{array}$$

and, Sub Problem-II:

$$EURM(2,) = \begin{array}{c} \\ t_1 \\ t_4 \end{array} \begin{array}{cc} p_2 & p_3 \\ 0.04 & 0.06 \\ 0.07 & 0.02 \end{array}$$

Step-4 & 5:

On applying modified Hungarian method devised by Kumar et al [KUMA 1995c] to assign the tasks, on the basis of min $\{r_i\}$ and min $\{c_j\}$ from unreliability matrices for every i and j. We put $r_{ij} = 0$ and $c_{ij} = 0$, for every i and j. On applying this to all sub problems, the modified matrices for each sub problem are mentioned below:

$$EURM(1,) = \begin{array}{c} \\ t_2 \\ t_3 \\ t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.07 & 0.00 & 0.08 \\ 0.00 & 0.09 & 0.06 \\ 0.08 & 0.05 & 0.00 \end{array}$$

$$EURM(2,) = \begin{array}{c} \\ t_1 \\ t_4 \end{array} \begin{array}{cc} p_2 & p_3 \\ 0.00 & 0.06 \\ 0.07 & 0.00 \end{array}$$

Step-6, 7&8:

After implementing assignment process, the allocation is thus obtained.

Tasks	→	Pr*ocessors*	*EUR*
t_1	→	p_2	0.04
t_2	→	p_2	0.02
t_3	→	p_1	0.02
t_4	→	p_3	0.02
t_5	→	p_3	0.04

Step-9:

EUR := 0.0000000128

Ereliability := 0.9999999872

Step-10:

Stop.

3.4.5 Conclusion

Maximizing the reliability of any system is one of the major parameter to enhance its performance. So that the present algorithm suggested here is capable for maximizing the overall reliability of communication system through task allocation. In these system tasks are allocated in such a way that their individual reliability of processing is optimized as well as it improve the overall reliability. In this approach not only that the loads of each processor get evenly balanced and none of the task gets unprocessed. This problem provide an optimal solution for assigning a set of "m" tasks of a program to a set of "n" processors where m > n for a communication system, that is to maximize the overall reliability of the system; the load of all allocated tasks on all the processors is evenly balanced. The execution unreliability and reliability of different processors has been obtained.

Processors	p_1	p_2	p_3
Execution unreliability of each processor PEUR (,)	0.02	0.0008	0.0008
Execution Unreliability [EUR]	0.0000000128		
Total Reliability [Ereliability(,)] = (1-EUR)	0.9999999872		

The final result of Example is as:

Tasks	→	Pr*ocessors*	*EUR*	*ER*
t_3	→	p_1		
$t_1 * t_2$	→	p_2	0.0000000128	0.9999999872
$t_4 * t_5$	→	p_3		

This problem provides the optimal solution for improving the reliability and also deals the performance on the bases of maximizing system reliability.

PROBLEM-V

OPTIMIZING THE EXECUTION TIME OF CS: THROUGH CLUSTERING

3.5.1 Objective

The main objective of this problem is to minimize the total program execution period by allocating the tasks in such a way that the allocated load on each processor should be evenly balanced. The model utilized the mathematical programming technique for execution of the module considering that each module to be executed through all the processors. The Execution Time and Inter Tasks Communication Time are considered for developing the algorithm. The impact of Inter Processor Distance also considered and it is mentioned in the Inter Processor Distance Matrix [IPDM (,)] of the order n. The ET and ITCT are presented by arrays in the form Execution Time

Matrix [ETM (,)] of order m x n and Inter Tasks Communication Time Matrix [ITCTM (,)] of order m.

3.5.2 Technique

Since the number of task are more than the number of processors, so that it is required to form the clusters of tasks. To forming the cluster of tasks arrange the upper diagonal value of ITCTM (,) in descending order and store the result in a linear array maxct() also store their respective positions in two dimensional array pos(,). The maximum number of tasks in a cluster shell be governed by the formula nt = [m/n], where nt is the number of tasks in a cluster. Select the first value of maxct() and its corresponding positions from pos(,) say, (t_i,t_k) and store the cl(i,j) where i= 1,2,..no. of cluster and j=1,2,..nt). if j<nt then pickup the next value from maxct() and its corresponding positions from pos(,) say, (t_l,t_k) select the task form this combination which not already exist in cl(i,j) say, t_l and store the same in cl(i,j). This process continues until the entire cluster to be formed. Some of the tasks, which are not involved in any cluster, are known as isolated tasks. Assign these clusters and isolated tasks by using the KSY Algorithm [KUMA 1995c]. Calculate the exaction time and inter tasks communication time with processor distance of each processor and store the result in a linear array pet(j) and pitctpd(j) respectively where j= 1,2,...n.

$$pet(i) = \sum_{j=1}^{m} e_{ij}\, x_{ij},\ i = 1,2,....n \left\{ \text{Where } x_{ij} = \begin{cases} 0, \text{ if } t_i \text{ and } t_j \text{ are on the same processor.} \\ 1, \text{ otherwise} \end{cases} \right\}$$

and

$$pitctpd(j) = \sum_{i=1}^{m} c_{ij} * d_{ij}\, x_i,\ j = 1,2,.....n \left\{ \text{Where } x_i = \begin{cases} 1, \text{ if } t_i \text{ is on the } j^{th} \text{ processor.} \\ 0, \text{ otherwise} \end{cases} \right\}$$

Finally, sum up the value of pet(j) and pitctpd(j), (j=1,….,n) and store the result the ttbp(j) and pickup the maximum value of ttbp(j) i.e. tostbp called as total optimal system time for busy period.

3.5.3 Computational Algorithm

The method discussed here is to determine the tasks allocations in communication systems based on the following components.

- Determine the initial allocation
- Determine the cluster of m-n tasks
- Determine the final allocation
- Computation the total system time of busy period.

3.5.4 Algorithm

To given an algorithmic representation to the technique mentioned in the previous section, let us consider a system in which a set T= $\{t_1, t_2, t_3 \ldots t_m\}$ of "m" tasks is to be executed on a set P= $\{p_1, p_2, p_3 \ldots p_n\}$ of "n" available processors.

Step-1: Input: m, n;//m is the number of modules of a task, n is the number of processor//

Input: etm(,);//matrix to hold the execution time of each task to each processor//

Input: itctm(,);//matrix to hold the Communication time amongst the tasks//

Step-2: maxct()←0;//linear array to store the inter task communication time//

i_{task}()←0;//linear array to store the tasks for initial assignment//

l_{task}()←0;//linear array to store the remaining tasks for initial assignment//

bmin←0;//Variable for selecting the best minimum//

T_{ass}()←0;//linear array to hold the tasks in order to assignment made//

$T_{non\text{-}ass}$()←0;//linear array to store the non assigned tasks//

nomade ←0;//variable for counting the number of the assignment made//

alloc()←0;//linear array to hold processor's position in order of assignment//

msr()←0;//linear array to hold processor mean service rate//

trp()←0;//linaer array to hold the throughput of the processors//

mst()←0;// linear array to hold processor mean service time //

pos(,)←0;//to dimensional array to hold to the corresponding position of maxct()//

cpos(,)←0;// to dimensional array to hold to the position of common element //

netm(,)←0;//operational matrix for execution time//

nictm(,)←0;// operational matrix for inter task communication time //

ttask()←0;//linear array to store the total number of task to the processors//

Step-3: for i ← 1 to n do

for j ← 1 to n do

store the etm(,) in netm(,) as

netm(,)← etm(,)

repeat

repeat

for i ← 1 to n do

for j ← 1 to n do

store the itctm(,) in nitctm(,) as

nitctm(,)← itctm(,)

repeat

repeat

Step-4: set k ← m (m-1)/2

Step-5: for i ← 1 to m do

for j ← 1 to k do

arrange the upper diagonal values of nitctm(,) in non–ascending order and store the result in maxct(), until j = k

repeat

repeat

Step-5.1: for i ← 1 to k do

for j ← 1 to 2 do

store the combinations of tasks of maxct() in pos(,)

repeat

repeat

Step-5.1.1: nt←0

set count ←m/n

Step-5.1.2: nt←count

for i ← 1 to m do

for j ← 1 to k do

pick-up the maximum value of maxct() and check the corresponding combination in pos(,), say(t_j,t_k)

if nt=count

then

store the cluster in a linear array cl_i()

else

if pos(i,j)<pos(i,k)

nmax ←pos(i,j), until j=k

repeat for i

nmax <pos(i,k), until j=k

endif

check the corresponding combination of nmax in pos(,),say(t_j,t_k), store the cluster in a linear array cl_i()

repeat

repeat

nt ←nt+count

Step-5.1.3: set k← (m-nt(m-nt)-1)/2

Step-5.1.4: modify the pos(,) by deleting the combination and reduce maxct() by eliminating the corresponding values.

modify the nectm(,) by adding the i^{th} and k^{th} rows together, also modify the nitctm(,) by adding the i^{th} and k^{th} rows and then column, remove k^{th} rows from nectm(,) and k^{th} column from nitctm(,)

goto Step-5.1.2.

Step-6: for k ← 1 to n do

for j ← 1 to n do

find out minimum of k^{th} rows, say mr_{kj}, of netm(i,j) lying in j^{th} column and subtract mr_{kj} from all the values of k^{th} rows

repeat

repeat

Step-6.1: for j ← 1 to n do

for k ← 1 to n do

find out minimum of j^{th} column, say mc_{kj}, of netm(i,j) lying in k^{th} row and subtract mc_{kj} from all the values of j^{th} column

repeat

repeat

Step-7: for k ← 1 to n do

for j ← 1 to n do

row in netm(,) has only one zero at position (1,2)

assign task t_1 to p_2;

allock(k) ←j; allock(2)

nomade←nomade+1; nomade=1

$t_{ass} \leftarrow t_{ass} \cup \{t_k\};\{t_1\}$

repeat

repeat

Step-7.1: for j ← 1 to n do

for k ← 1 to n do

search for a column in netm(,), which has only one zero, say at position (k,j); assign task(s) corresponding to k^{th} , say, t_k, row to p^{th} processor, say p_j,

allock(k) ←j

nomade←nomade+1

$t_{ass} \leftarrow t_{ass} \cup \{t_k\}$

repeat

repeat

Step-8: if nomade≠ n

then

pick-up an arbitrary zero,

go to Step–7

else

go to Step-7.1

endif

Step-8.1: check column(s) position of zero(s) in unassigned row(s) check the row (s) any previous assignment in the corresponding column(s) store the positions of the common elements in cpos(,), say (i,j), find the minimum element of all the elements of the remaining rows, say min_{ij}, subtract min_{ij}, from these elements add min_{ij}, at the common positions and then go to Step-6.

Step-9: for k ← 1 to m do

for j ← 1 to n do

compute the et_{ij} by summing-up th value of etij for each processors and store the result in a linear array pet(j)

compute the mean service rate of the p^{th} processor, say p_j, stored in allock(k) for the assigned tasks corresponding to k^{th}, say t_k, in $t_{ass}()$

msr(j)←1/pet(j)

repeat

repeat

Step-9.1: for j ← 1 to n do

Compute the maen services time of the processors and store the result in a linear array mst(j)

mst(j)←1/msr(j)

repeat

Step-9.2: for i ← 1 to n do

pcount←0

for j← 1 to m do

compute the processor's throughput and store the result in a linear array trp(i) as,

if i = allock(j)

then

pcount← pcount+1

else

```
                                        next j
                                    endif

                                    ttask(i) ←pcount

                                repeat

                                trp(i) ←ttask/mst(i)

                            repeat

Step-10:        tcc←0

                            for i ← 1 to m do

                                for j ← 1 to n do

                                compute the itct for each processor as,

                                pcc()←tcc+itctm(i,j)

                            repeat
                    repeat

Step-11: for j ← 1 to n do

                            compute the total busy time for each processor

                            tpbt()←pet()+pct()

                    repeat

                    and select the maximum value from tbpt()

                    tost←max{tpbt()}//tost is the total system time

Step-12: Stop.
```

3.5.5 Implementation

Consider a communication system which is consisting of a set P = {p_1, p_2, p_3} of "n = 3" processors connected by an arbitrary network. The processors only have local memory and do not share any global memory. A set T = {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8,} of "m = 8" executable tasks which may be portion of an executable code or a data file.

$ipdm(,) =$

	p_1	p_2	p_3
p_1	0	4	2
p_2	4	0	3
p_3	2	3	0

etm(,) =

	p_1	p_2	p_3
t_1	6	3	5
t_2	4	2	3
t_3	3	1	2
t_4	5	2	∞
t_5	3	4	2
t_6	6	∞	6
t_7	5	6	7
t_8	∞	2	5

itctm(,)=

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1	0	3	4	2	6	8	1	0
t_2	3	0	0	0	0	0	0	5
t_3	4	0	0	4	3	2	0	0
t_4	2	0	4	0	5	3	2	5
t_5	6	0	3	5	0	0	0	0
t_6	8	0	2	3	0	0	6	8
t_7	1	0	0	2	0	6	0	5
t_8	0	5	0	5	0	8	5	0

$$Cluster \quad = cl\ (i,\ j) = \begin{bmatrix} 1 : t_1, t_6, t_8 \\ 2 : t_3, t_4, t_5 \end{bmatrix}$$

tasks t_2 and t_7 are not involved in any cluster known as isolated tasks. After applying the KSY [KUMA 1995c] algorithm the final allocations are

Tasks	Processor	Time taken by the tasks for execution on particular processor
t_1	P_3	5
t_2	P_1	4
t_3	P_2	1
t_4	P_2	2
t_5	P_2	4
t_6	P_3	6
t_7	P_1	5
t_8	P_3	5

Execution time of each processor	pet()	=	(9,	7, 16)
Inter tasks communication time with inter-processor distance of each processor	pitctpd()=		(48,	74, 106)
Total time of busy period of each processor	ttbpt()	=	(57,	81, 122)
Maximum value of ttbp()		=	122	
Tostbp (i.e. total optimal system time for busy period)		=	122	

3.5.6 Conclusion

The model discussed here provides an optimal solution in a Communication System to maximize the overall throughput and the balanced load of all allocated tasks on each processor. This approach forms the clusters before making the assignments in the example mentioned in the implementation. Two clusters have been formed of containing three tasks. There is two tasks are not involved in any of the cluster treated as isolated tasks. The Inter Task Communication Time with Inter

Processor Distance and Execution Time on different processors has been obtained. The total optimal system cost for busy period in the example mentioned in the body of the chapter is found 122 units. The final results are shown in the following table:

Processor → Tasks	Pet()	Pitctpd()	ttbpt()	Tostbp
$P_1 \rightarrow t_2, t_7$	9	48	57	122
$P_2 \rightarrow t_3, t_4, t_5$	7	74	81	
$P_3 \rightarrow t_1, t_6, t_8$	16	106	122	

The graphical representation of the optimal assignment has been shown in the figure-4. Out of two clusters one goes to processor p_2 while the other to processor p_3. The isolated tasks are executed on processor p_1. The total optimal system time for busy period is 122 units which include the impact of the processor distance and inter tasks communication time. The developed model is programmed in C++ and implemented the several sets of input data are used to test the effectiveness and efficiency of the algorithm. It is found that the model is suitable for arbitrary number of processors with the random program structure.

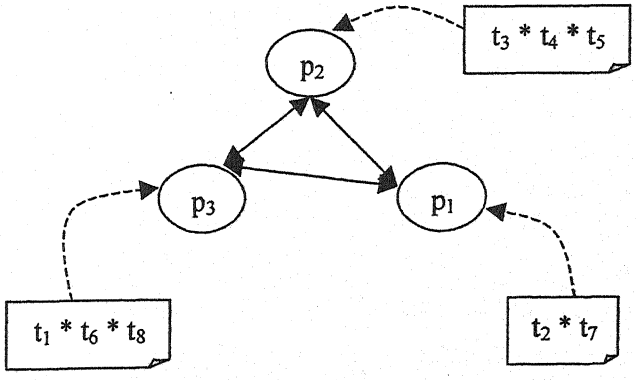


Figure 4: Optimal Assignment Graph

CHAPTER-4

TECHNICAL PUBLICATIONS AND SUGGESTIONS FOR FUTURE RESEARCH

4.1. TECHNICAL PUBLICATIONS

1. A research paper entitled, **"Evaluation of Improved Reliability of Distributed Computing Systems using Mathematical Programming"** Published GAMS Journal of Mathematical Bio Sciences, Vol. 3 (1) pp. 88-96, 2007
2. A research paper entitled, **"Evaluation of Optimal execution time of Computer System by Exhaustive Search Approach"** Research Link –32 Vol-V(7) pp. 12-15 September-November 2006
3. A research paper entitled, **"Evaluation of Optimal Time of Computer system Using Exhaustive Search Approach"** presented at the National Seminar on Mathematics and Computer Sciences at Muzaffarnagar from 29-30 Nov.2005.
4. A research paper entitled, **"Evolving the Strategy for Optimizing the time of Distributed Computing System using Matrix Partitioning Approach"** presented at the Seminar on Current trends in Mathematics and Computation at BIT, Noida on 01-12-2005.
5. A research paper entitled, **"Evaluation of Improved Reliability of Distributed Computing Systems using Mathematical Programming"** presented at the 11th Annual Conference of Gwalior Academy of Mathematical Sciences and National Symposium on Applicable Mathematics to Engineering and Technology, Raghogarh (Guna) from 22-23 April 2006.
6. A research paper entitled, **"Evolving the Strategy for Optimizing the time of Distributed Computing System using Matrix Partitioning Approach"** communicate to RdE Journal of Mathematical Sciences.

4.2. FUTURE SCOPE

The present research work entitled "TO STUDY AND ANALYSIS OF OPTIMIZATION TECHNIQUES AND THEIR APPLICATION TO EVOLVE SOME ALGORITHM FOR PERFORMANCE ENHANCEMENT OF COMMUNICATION SYSTEM" is devoted to the optimization techniques, communication systems, task allocation and other directly related to the issues. Task assignment of any communication system is a most computing interesting and demandable research problem. Various methodologies and techniques are available in the literature to solve such problems. Keeping in view, the importance and necessity of performance evaluation of communication system, the future aim of study would be to develop some algorithms and techniques regarding the following problems.

- Developing new techniques & their algorithm for minimizing the delays among various processors of communication systems.
- Developing new techniques & their algorithm for proper utilization of the processors of the communication system to avoid overloading the processors.
- Developing the algorithm for assignment.
- To devise some approaches to evaluation of the performance of communication system.
- Developing some of the dynamic algorithms for assignment policy and its implementation on various fields of daily life.
- Developing the criterion for forming the clusters has to be taken into consideration and its implementation.
- Use of genetic algorithm may provide some interesting solutions for such types of problems.
- Use of neural algorithm may also provide some good solutions for such types of problems.

REFERENCES

[AGAR 1995] Aggarwal, D. C., "Computer Communication and ISDN Systems", Khanna Publishers, New Delhi, 1995.

[AGGA 1982] Aggarwal, K.K., Chopra, Y. C., and Bajwa, J. S., "Reliability Evaluation by Network Decomposition", IEEE Transactions on Reliability, Vol. R-31, No. 4, 1982, pp. 355-358.

[ALFO 1988] Alford, M.W., "Heuristic Algorithm for Task Assignment in Distributed Systems", IEEE Transaction on Computers, Vol. 37, No. 11, 1988, pp. 1384-1397.

[AROR 1979] Arora, R. K., and Rana, S. P., "On Module Assignment in Two Processors Distributed Systems", Information Processing Letters, Vol. 9, No. 3, 1979, pp. 113-117.

[AROR 1980] Arora, R. K., and Rana, S. P., " Heuristic Algorithms for Process Assignment in Distributed Computing Systems", Information Processing Letters, Vol. 11, No. 45, 1980, pp. 199-203.

[AROR 1981] Arora, R. K., and Rana, S. P., "On the Design of Process Assigner for Distributed Computing Systems", The Australian Computer Journal, Vol. 13, No. 3, 1981, pp. 77-82.

[ASHR 1992] Ashrafi, N., and Berman, O., "Optimization Models for Selection of Programs, Considering Cost and Reliability", IEEE Transactions on Reliability, Vol. 41, No. 2, 1992, pp. 281-287.

[BACA 1989] Baca, D.F., "Allocation Modules to Processor in a Distributed Systems", IEEE Transactions on Software Engineering, Vol. SE-15, 1989, pp. 1427-1436.

[BERG 1985] Berger, M. J., and Bokhari, S. H., " A Partitioning Strategy for PDEs Across Multiprocessors", Proc. of the 1985, International Conference on Parallel Processing, 1985, pp. 166-170.

[BERG 1987] Berger, M. J., and Bokhari, S. H., "A Partitioning Strategy for Non Uniform Problems on Multiprocessors", IEEE Transactions on Computers, Vol. C-36, No. 5, 1987, pp. 570-580.

[BERM 1984] Berman, F., and Synder, L., "On Mapping Parallel Algorithms into Parallel Architecture", Proceeding of the International Conference on Parallel Processing, 1984, pp. 307 - 309.

[BHAR 1994] Bharadwaj, V., Ghosh, D., and Mani, V., "Optimal Sequencing and Arrangement in Distributed Single-Level Tree Network with Communication Delays", IEEE Transactions on Parallel & Distributed Systems, Vol.5, No. 9, 1994, PP. 968-976.

[BHUT 1994] Bhutani, K.K., "Distributed Computing", The Indian Journal of Telecommunication, 1994, PP. 41-44.

[BIER 2002] Bierbaum L. Rene, Brown D. Thomas, and Kerschen, J., Thomas., "Model Based Reliability Analysis", IEEE Transactions on Reliability, Vol. 51, No. 2, 2002, pp. 133-140.

[BOKH 1979] Bokhari, S. H., "Dual Processor Scheduling with Dynamic Reassignment", IEEE Transactions on Software Engineering, Vol. SE-5, 1979, pp. 341-349.

[BOKH 1981a] Bokhari, S. H., " A Shortest Tree Algorithm for Optimal Assignment Across Space and Time in Distributed processor System", IEEE Transactions on Software Engineering, Vol. SE-7, No. 6, 1981, pp. 583-589.

[BOKH 1987] Bokhari, S.H., "Assignment Problems in Parallel and Distributed Computing", Kluwer Academic Publisher, 1987.

[BOKH 1988] Bokhari, S.H., "Partitioning Problems in Parallel, Pipeline and Distributed Computing", IEEE Transactions on Computers, Vol. C-37, No. 1, 1988, pp. 48-57.

[BOKH 1993] Bokhari, S.H., "A Network Flow model for Load Balancing in Circuit-Switched Computer", IEEE Transactions on Parallel & Distributed Systems, Vol. 4, No. 6, 1993, PP. 649-657.

[BOLC 1983] Bolch, G., Hofmann, F., Hoppe, B., Kolb, H.J., Linster, C.U., Polzer, R., Schussler, W., Wackersreuther, G., and Wurm, F.X., "A Multi Processor System for Simulating data Transmission System (MUPSI)", Micro-processing and Microprogramming, Vol. 12, 1983, pp. 267-277.

[BOWE 1992] Bowen, N. S., Nikolaou, C. N., and Ghafoor, A., "On the Assignment Problem of Arbitrary Process Systems to Heterogeneous Distributed Computer Systems," IEEE Transactions on Computers, Vol. 41, No. 3, 1992.

[BUCK 1979] Buckels, B.P., and hardin, D.M., "Partitioning and Allocation of Logical Resources in a Distributed Computing Environment', Tutorial: Distributed System Design", IEEE Computer Society Press, 1979.

[CASA 1988] Casavant, T. L., and Kuhl, J. G., "A Taxonomy of Scheduling in General Purpose Distributed Computing System", IEEE Transactions on Software Engineering, Vol. SE-14, 1988, pp. 141-154.

[CHIN 2006] Chin-Ching Chiu, Chung-Hsien Hsu, Yi-Shiung Yeh, "A Genetic Algorithm for Reliability-Oriented Task Assignment with k/spl tilde / duplication in Distributed Systems", IEEE Transactions on Reliability, Vol. 55, No. 1, 2006, PP. 105-117.

[CHOU 1986] Chou, T.C.K., and Abraham, J.A., "Distributed Control of Computer Systems", IEEE Transactions on Computers, Vol. C-35, No. 6, 1986, PP. 564-567.

[CHU 1969] Chu, W. W., "Optimal File Allocation in a Multiple Computing System", IEEE Transactions on Computers, Vol. C-18, 1969, pp. 885-889.

[CHU 1980a] Chu, W.W., Holloway, L. J., Lan, L. M. T., and Efe, K., "Task Allocation in Distributed Data Processing", IEEE Transactions on Computers, Vol. 13, No. 11, 1980, pp. 57-69.

[CHU 1980b] Chu, W.W., "Introduction in the Special Issue on Distributed processing", IEEE Transactions on Computers, Vol. C-29, 1980, pp. 1037-1038.

[CHU 1984b] Chu, W. W., and Leung, K.K., "Task Response Time Model and its Application for Real Time Distributed Processing Systems", Proceeding of 5th Real-Time Systems Symposium Texas, 1984, PP.225-236.

[CHU 1987b] Chu, W. W., and Lan, L.M.T., "Task Allocation and Precedence Relation for Distributed Real-Time", IEEE Transactions on Computers, Vol. C-36, No.6, 1987, PP.667-679.

[CHUA 1992] Chuang, Po-jen, and Tzeng, Nian Feng, " A Fast Recognition Complete Processor Allocation Strategy Hypercube Computer", IEEE Transactions on Computers, Vol. 41, No. 4, 1992, pp. 467-479.

[COIT 1996] Coit, W., David, and Smith, A. E., "Reliability Optimization of Series-Parallel Systems using Genetic Algorithm", IEEE Transactions on Reliability, Vol. 45, No. 2, 1996, pp.254-266.

[COIT 1998a] Coit, W., David, and Smith, A. E., "Redundancy Allocation to Maximize a Lower Percentile of the System Time-to-Failure Distribution", IEEE Transactions on Reliability, Vol. 47, No. 1, 1998, PP. 79-87.

[COIT 1998b] Coit, D. W., "Economic Allocation of Test Times for Sub-System level Reliability Growth Testing," IEEE Transaction on Reliability, Vol. 30, No. 12, 1998, pp.1143-1151.

[COST 2007] Costa and E.O. de Souza, G.A., Pozo, A.T.R., Vergilio, S.R., "Exploring Genetic Programming and Boosting Techniques to Model Software Reliability," IEEE Transaction on Reliability, Vol. 56, No. 3, 2007, pp.422-434.

[DENG 1997] Dengiz, Benna, Altiparmak, Fulya, and Smith Alice, E., "Efficient Optimization of All-Terminal Reliable Networks,

Using and Evolutionary Approach", IEEE Transactions on Reliability, Vol. 46, No. 1, 1997, pp. 18-26.

[DESS 1980] Dessoukiu-EI, O.I., and Huna, W.H., "Distributed Enumeration on Network Computers," IEEE Trans. On Computer, Vol.C-29 pp.818-825, 1980.

[DINI 1970] Dinic, E.A., "Algorithm for Solution of a Program of Maximum Flowing a Network with Power Estimation", Soviet Math. Doklady, Vol. 11, No. 5, 1970, pp. 1277-1280.

[EDMO 1972] Edmonds, J., and Karp, R. M., "Theoretical Improvements in Algorithms Deficiency for Network Flow Problems", J. Asso. Comp. Mech., Vol. 19, 1972, pp. 248-264

[EFE 1982] Efe. K., " Heuristic Models of task Assignment Scheduling in Distributed Systems", IEEE Transactions on Computers, Vol. 5, No. 6, 1982, pp. 50-56.

[FABO 1976] Fabozzi, F.J., and Valaente, J., "Mathematical Programming in American Companies : A sample survey", Interfaces, Vol. 7, 1976, PP. 93-98.

[FERN 1989] Fernandez-Baca, David, "Allocating Modules to Processors in a Distributed Systems", IEEE Transactions of Software Engineering, Vol. SE-15, No. 11, 1989, pp. 1427-1436.

[FITZ 1988] Fitzserald, J., "Business Data Communication", John Wiley & Sons, New York, 1988.

[FITZ 2002] Fitzserald, Kent, Latifi, Shahram, Srimani, Pradeep, K., "Reliability Modeling and Assessment of the Star Graph Network", IEEE Transactions of Reliability, Vol. 51, No. 1, 2002, pp. 49-57.

[FORD 1962] Ford, L. R. Jr., and Fulkerson, D. R., " Flows in Networks", Princeton University Press, 1962.

[FORT 1985] Forter, P.J., "Design and Analysis of Distributed Real-Time System", Inter Text Publication Inc., and McGraw-Hill, Inc., New York, 1985.

[FUKU 1987] Fukunage, K., Yamada, S., and Kasail, T., "Assignment of Job Modules on to Array Processors", IEEE Transactions on Computers, Vol. C-36, No. 7, 1987, pp. 881-891.

[GARC 1982] Garcia-Molina, J., "Reliability Issues for Fully Replicated Database", IEEE Transactions on Computers, Vol. 16, 1982, PP. 34-42.

[GILL 2002] Gillett, Billy, E., "Introduction to Operation Research-A Computer Oriented Algorithmic Approach", Tata McGraw-Hill Publishing Company Limited, New Delhi, 2002

[GRAY 1973] Grayson, C.J., "Management Science and Business Practice", Howard Business Review, Vol. 51, 1973, PP. 41-48.

[HA 2006] Ha, C and Kuo, W.,"Multi-path Heuristic for Redundancy Allocation : The Three Heuristic", IEEE Transaction on Reliability, Vol. 55, No. 1, 2006, pp. 37-43.

[HOLT 1978] Holt, A. W. et al., "Final Report of the Information System Theory Project", Room Air Development Center, No. AD, 1978, pp. 676- 972.

[HUI 1997] Hui, C.C., and Chanson, S.T., "Allocation Task Interaction Graphs to Processors in Hetrogeneous Networks", IEEE Transactions on Parallel and Distributed System, Vol.8, No.9, 1997, PP. 908-915.

[HWAN 1993] Hwang, G., and Teeng, S., "A Heuristic Task Assignment Algorithm to Maximize Reliability of Distributed System", IEEE Transactions on Reliability, Vol. 42, No. 3, 1993, pp. 408-415.

[IQBA 1986a] Iqbal, M.A., "Approximate Algorithms for Partitioning and Assignment Problems", ICASE Report No. 86-40, 1986

[IQBA 1986b] Iqbal, M.A., Saltz, J.H., and Bokhari, S.H., "A Comparative Analysis of Static and Dynamic Load Balancing Strategies", Proceeding of the 1986 International Conference on Parallel Processing, 1986, pp. 1040-1047.

[JOLL 1980] Joller, J. M., "Reliability Block Diagrams and Petri Net", Microelectronics and Reliability, Vol. 20, 1980, pp.613-624.

[KART 1995] Kartik, S., and Murthy, C. Siva Ram, "Improved Task-Allocation Algorithms to Maximize Reliability of Redundant Distributed Computing Systems", IEEE Transactions on Reliability, Vol. 44, No. 4, 1995, pp. 575-586.

[KARZ 1974] Karzanova, A. V., "Determining the Maximum Flow in Network by Method for Preflows", Soviet Math. Doklady, Vol. 15, No. 2, 1974, pp. 434-437.

[KE 1997] Ke, Wei-Jehn, and Wang, Sheng-De, "Reliability Evaluation for Distributed Computing Networks with Imperfect Nodes", IEEE Transactions on Reliability, Vol. 46, No. 3, 1997, pp. 342-349.

[KEIN 1971] Keingham, B.W., "Optimal Sequential Partition of Graphs", Journal of the ACM, Vol. 18, No. 1, 1971, PP. 34-40.

[KIM 1988] Kim, S. J. and Browne, " A General Approach to Mapping of Parallel Computations upon Multiprocessor Architectures", Proceeding International Conference on Parallel Processing, Vol. – 3, Aug 1998, pp. 1-8.

[KOHL 1975] Kohler, W.H.,"A Perliminary Evaluation of the Critical Path Method for Scheduling Tasks on Multiple Processor Systems," IEEE Trans. Computer, vol. 24, no. 12, pp. 1,235-1,238, Dec. 1975.

[KUMA 1988] Kumar, Vinod, and Aggarwal, K.K., "Determination of Path Identifiers for Reliability Analysis of a Broadcasting Network Using Petri Nets", International Journal of Systems Sciences, Vol. 19, No. 12, 1988, pp. 2643-2653.

[KUMA 1990] Kumar, Vinod, and Aggarwal, K.K., "Efficient Computerized Petri Net Approach for the Enumeration of the Sets of Path Identifier for Reliability Analysis of Broadcasting Network", International Journal of Systems Sciences, Vol. 21, No. 7, 1990, pp. 1239-1248.

[KUMA 1993] Kumar, Vinod, and Aggarwal, K.K., "Petri Net Modeling and Reliability Evaluation of Distributed Processing Systems", Reliability Engineering and Systems Safety, Vol. 41, 1993, pp. 167-176.

[KUMA1995a] Kumar, A., Pathak, R. M., Gupta, Y. P., and Parsaei H.R., "A Genetic Algorithm for Distributed System Topology Design", Computers and Industrial Engineering, Vol. 28, 1995, pp. 659-670.

[KUMA1995b] Kumar, A., Pathak, R. M., Gupta, Y. P., "Genetic Algorithm based Reliability Optimization for Computer Network Expansion", IEEE Transactions on Reliability, Vol. 44, No. 1, 1995, pp. 63-72.

[KUMA 1995c] Kumar, Vinod, Singh, M.P., and Yadav, P.K., "A fast algorithm for allocation task in distributed processing system," proceedings of the 30th Annual convention of computer society of India, Hyderabad, pp.347-358, 1995.

[KUMA 1995d] Kumar, Vinod, Singh, M.P., and Yadav, P.K., "An efficient Algorithm for allocating tasks to processors in a distributed system," Proceedings of the Nineteenth National system conference (NSC-95), PSC College of technology, Coimbatore, organized by system society of India, New Delhi, pp. 82-87, 1995.

[KUMA 1996] Kumar, Vinod., Singh, M.P., and Yadav, P.K.,"An Efficient Algorithm for Multi-processor Scheduling with Dynamic Re-Assignment", Proceeding of the 6th National Seminar on Theoretical Computer Science. Held at Banasthali Vidyapith, India, pp. 105-118, 1996.

[KUMA 1998] Kumar, Vinod, Yadav, P.K., and Bhatia, K.,"Optimal Task Allocation in Distributed Computing Systems Owing to Inter Task Communication Effects." Proc. Of CSI-98 : IT for the New Generation, pp.369-378, 1998.

[KUMA 1999] Kumar, Avanish, "Optimizing for the Dynamic Task Allocation", Published to the proceedings of the III Conference of the International Academy of Physical Sciences held at Allahabad, 1999, pp. 281-294.

[KUMA 2001] Kumar, Avanish, "An Algorithm for Optimal Index to Task Allocation Based on Reliability and Cost", Published to the proceedings of International Conference on Mathematical Moideling held at Roorkee, 2001, pp. 150-155.

[KUMA 2002] Kumar, Avanish, and Yadav, P.K., "An Efficient Static Approach for Allocation Through Reliability Optimization in Distributed Systems", Presented at the International Conference on Operation Research for Development (ICORD2002) Held at Chennai.

[KUMA 2006] Kumar, Harendra, Singh, M.P., Kumar, Avanish, "A Task Allocation Model for Distributed Data Network", Journal of Mathematical Sciences, Vol.1, no.4, 2006, pp.379-392.

[KWAK 1987] Kwak, N.K., and Moor, J.S., "Introduction to Mathematical Programming", Robert E. Krieger Publishing Company Manabar, Florida. 1987.

[LAN 1985] Lan, L. M. T., "Characterization of Inter Module Communications and Heuristic Task Allocation for Distributed Real -Time System", Ph.D. Dissertation, Computer Science Department University of California, Los Angeles, 1985.

[LEDB 1977] Ledbetter, W.N., and Cox, J.F., "Are Or Techniques Being Used?", Industrial Engineering, Vol. 27, 1977, PP. 19-21.

[LEE 1987] Lee, S. Y., and Aggarwal, K.K., "A Mapping Strategy for Parallel Processing", IEEE Transactions on Computers, Vol. C-36, No. 4, 1987, pp. 433-441.

[LEVI 1998] Levitin, Gregory, Lisnianski, Anatoly, Ben-Haim, Hanoch, and Elmakis, David, "Redundancy Optimization for Series-Parallel

Mult-State System", IEEE Transactions on Reliability, Vol. 47, No. 1, 1998, pp. 165-172.

[LI 1992] Li, Duan, and Haimes, Y. Y., " A Decomposition Method for Optimization of Large–system Reliability", IEEE Transactions on Reliability, Vol. 41, No. 2, 1992, pp. 183-188.

[LIAN 1994] Liang, Ting-Peng, Lai, H., Chen, Nain-Shing, Wai, H., and Chen, M. C., "When Client/Server Isn't enough: Coordinating Multiple Distributed Tasks", IEEE Computers, Vol. 27, No. 5, 1994, pp. 73-79.

[LIN 1990] Lin, F.T., and Hsu, C.C.," Task Assignment Scheduling by Simulated Annealing," IEEE Region 10 Conference on Computer and Communication Systems, pp.279-283, Hong Kong, September 1990.

[LIN 2002] Lin, Min – Sheng, "A Linear Time Algorithm for Computing K-Terminal Reliability on Proper Interval Graph" IEEE Transactions on Reliability, Vol. 51, No. 1, 2002, pp. 58-62.

[LINT 1981] Lint, B., and Agarwal, T., "Communication Issues in the Design and Analysis of Parallel Algorithm", IEEE Transactions on Software Engineering, Vol. SE-7, No. 2, 1981, pp. 174-188.

[LIU 1999] Liu, Xiaoming, Kreitz, Christoph, Renesse, Robbert van, Hickey, Jason, Hayden, Mark, Kenneth, Birman, Constable, Robert, "Building reliable, high-performance communication systems from components", ACM Symposium on Operating Systems Principles, Charleston, South Carolina United States, 1999, pp. 80-92.

[LO 1988] Lo, V.M., "Heuristic Algorithms for Task Assignments in Distributed Systems", IEEE Transactions on Computers, Vol. 37, No. 11, 1988, pp. 1384-1397.

[LOPE 1994] Lopez-Benitez, N., "Dependability Modeling and Analysis of Distributed Programs", IEEE Transactions on Software Engineering, Vol. 20, No. 5, 1994, pp. 345-352.

[LU 1986] Lu, H., and Carey, M. J., "Load Balanced Task Allocation in Locally Distributed Computer Systems", Proceeding of the 1986 International Conference on Parallel Processing, 1986, pp. 1037-1039.

[LYU 2002] Lyu, Michael, R., Rangarajan, Sampath and Moorsel, and P.A. Van, "Optimal Allocation of Test Resources for Software Reliability Growth Modeling in Software Development", IEEE Transactions on Reliability, Vol. R-51, 2002, pp. 183-192.

[MA 1982] Ma, P. Y. R., Lee, E. Y. S., and Tuschiya, M., "A Task Allocation Model for Distributed Computing Systems", IEEE Transactions on Computers, Vol. C-31, No. 1, 1982, pp. 41-47.

[MA 1984] Ma, P. Y. R., "A Model to Solve Timing Critical Application on Distributed Computer Systems", IEEE Transactions on Computers, Vol. 17, No. 1, 1984, pp. 62-68.

[MANI 1998] Manimaran, G., Murthy, C.S.R., "A Fault – Tolerant Dynamic Scheduling Algorithm for Multi-processor Real – time Systems and its Analysis," IEEE Transactions on Parallel and Distributed Systems, Vol. 9, No. 11, Nov. 1998, pp. 1137-1152.

[MARC 1981] Marcogliese, R., and Novarese, R., "Module and Data Allocation Methods in Distributed Systems", Proceeding of the 1981 International Conference on Distributed Computer Systems, 1981, pp. 50-59.

[MARK 1972] Markland, R.E., and Newett, R.J., "A Subjective Taxonomy for Evaluation the Stages of Management Science", Interfaces, Vol.2, 1972, PP. 31-39.

[MARK 1986] Markenscoff, P., and Liew, W., "Task Allocation Problems in Distributed Computer System", Proceeding of the 1986 International Conference on Parallel Processing, 1986, PP. 953-960.

[MART 1988] Martin, J., "Computer Network and Distributed Processing (Software Techniques and Architecture)", Prentice Hall of India Pvt. Ltd., Delhi, 1988.

[MEND 1987] Mendelson, B., and Silberman, G.M., "An Improved Mapping of Data Flow Programs on a VLSI Array of Processor", Proceeding of the 1987 International Conference on Parallel Processing, 1987, pp. 871-874.

[MURA 1979] Murata, T., "Relevance of Network Theory to Models of Distributed Processing", Proceeding 1979 International Colloquium Circuits systems, Japan, 1979.

[MURT 1988] Murthy, C., Siva Ram, and Rajaraman, V., "Multiprocessor Architecture for Solving PDE's", Journal of the Institution of Electronics and Telecommunication Engineers, Vol. 34, No. 3, 1988, pp. 172-184.

[MURT 1989] Murthy, C., Siva Ram, and Rajaraman, V., "Task Assignment in a Multiprocessor System", Micro Processing and Microprogramming, Vol. 26, 1989, pp. 63-71.

[NAKA 1976] Nakazawa, H., "Bayesian Decomposition Method for Computing the Reliability of an Oriented Network", IEEE Transactions on Reliability, Vol. R-25, No. 2, 1976, pp. 77-80.

[NAKA 1977] Nakazawa, H., " A Decomposition Method for Computing System Reliability by a Boolean Expression", IEEE Transactions on Reliability, Vol. R-26, No. 4,1977, pp. 250-252.

[NAKA 1978] Nakazawa, H., " A Decomposition Method for Computing System Reliability by a Matrix Expression", IEEE Transactions on Reliability, Vol. R-27, No. 5, 1978, pp. 342-344.

[NAKA 1981] Nakazawa, H., "Decomposition Method for Computing the Reliability of Complex Networks", IEEE Transactions on Reliability, Vol. R-30, No. 3, 1981, pp.289-292.

[OLSO 1994] Olson, A., and Shin, K. G., " Fault-Tolerant Routing in Mesh Architectures", IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 11, 1994, pp. 1225-1232.

[ONIS 2007] Onishi J., Kimura, S., James, R.J.W., Nakagawa, Y., "Solving the Redundancy Allocation Problem with a Mix of Components using the Improved Surrogate Constraint Method, " IEEE Transaction on Reliability, Vol. 56, No. 1, 2007, pp. 94-101.

[PAIN 1995] Painton, L., and Compbell, J., "Genetic Algorithms in Optimization of System Reliability", IEEE Transactions on Reliability, Vol. 44, No. 2, 1995, pp. 172-178.

[PENG 1997] Peng, Dar-Tezen, Shin, K.G., and Abdel Zoher T.F., "Assignment Scheduling Communication Periodic Task in Distributed Real Time System", IEEE Transactions on Software Engineering, Vol. SE-13, 1997, pp. 745-757.

[PETR 1962] Petri, C.A., "Communication with Automata", Transactions C. P. Greene, Jr, (Kommunikation Mit Automatcn), University of Bonn, 1962.

[PRIC 1982] Price, C.C., and Pooch, U.W., "Search Techniques for Non-Linear Multiprocessor Scheduling Problems", Naval Research Logistics Quarterly, Vol. 29, No. 2, 1982, PP. 213-233.

[PRIC 1984] Price, C.C., and Krishnaprasad, S., "Software Allocation Models for Distributed Computing Systems", Proceeding of the 4th International Conference on Distributed Computing Systems, 1984, pp. 40-48.

[RADN 1973] Radnor, M., and Neal, R.D., "The Process of Management Science Activities in Large U.S. Industrial Corporation", Operation Research, Vol. 21, 1973, PP. 427-450.

[RAGH 1985] Raghavendra, C. S., and Hariri, S., "Reliability Optimization in the Design of Distributed Systems", IEEE Transactions on Software Engineering, Vol. SE-11, 1985, pp. 1184-1193.

[RAI 1982] Rai, Suresh, "A Cut- Set Approach to Reliability Evaluation in Communication Networks", IEEE Transactions on Reliability, Vol. R-31, No. 5, 1982, pp. 428-431.

[RAMI 2006] Ramirez-Marquiz, J.E. and Wei, Jian.,"On Improved Confidence Bounds for System Reliability," IEEE Transaction on Reliability, Vol. 55, No. 1, 2006, pp. 26-36.

[RAMI 2007] Ramirez–Marquez, J.E. and Gebre, B.A., "A Classification Tree Based Approach for the Development of Minimal Cut and Path Vectors of a Capacitated Network," IEEE Transactions on Reliability, Vol. 56, No. 3, 2007, pp. 474-487.

[REEV 1993] Reeves, C. R., "Modern Heuristic Techniques for Combinational Problems," John Wiley & Sons, 1993.

[REID 1994] Reid, D. J., "Optimal Distributed Execution of Join Queries", Computer Math. Application, Vol. 27, No. 11, 1994, pp. 27-40.

[RICH 1982] Richard, R.Y., Lee, E.Y.S., and Tsuchiya, M., "A Task Allocation Model fro Distributed Computer System", IEEE Transactions on Computers, Vol. C-31, 1982, pp. 41-47.

[ROSE 1982] Rosenthal, A., "Dynamic Programming is Optimal for Non Serial Optimization Problems", SIAM, journal of Computer, Vol. 11, No. 1, 1982, pp. 47-59.

[ROTI 1994] Rotithor, H.G., "Taxonomy of Dynamic Task Scheduling Schemes in Distributed Computing Systems", IEEE Proceedings-Computer Digital Techniques, Vol. 141, No. 1, 1994, PP. 1-10.

[SAGA 1991] Sagar, G., and Sarje, A. K., "Task Allocation Model for Distributed System", International Journal of Systems Sciences, Vol. 22, No. 9, 1991, pp. 1671-1678.

[SCHN 2007] Schneeweiss, W.G., "Review of Petri Net Picture Book" and "Petri Nets for Reliability Modeling.", IEEE Transaction on Reliability, Vol. 55, No. 2, 2006, pp. 391-392.

[SEGE 1994] Segee, B.E., and Carter, M.J., "Comparative Fault Tolerance of parallel Distributed Processing networks", IEEE Trans. on Computer, Vol. 43, No.11, 1994, pp. 1323-1329.

[SHAT 1987] Shatz, S.M., and Wang, Jai-Ping, "Introduction to Distributed Software Engineering", Computers, Vol. 20, No. 10, 1987, pp. 23-30.

[SHAT 1989] Shatz, S.M., and Wang, Jai-Ping, " Models and Algorithms for Reliability-Oriented Task, Allocation in Redundant Distributed Computer Systems", IEEE Transactions on Reliability, Vol. 38, No. 1, 1989, pp. 16-27.

[SHAT 1992] Shatz, S. M., and Wang, Jai-Ping, " Task Allocation for Maximizing Reliability of Distributed Computer Systems", IEEE Transactions on Computers, Vol. 41, No. 9, 1992, pp. 1156-1168.

[SHMI 1991] Shmilovici, A., and Maimon, O. Z., "Estimating the Performance of Multi-process, Distributed Processing System that is Controlled with Fixed Priorities", 17th Convention of Electrical and Electronics Engineers in Israel (Proceedings), No. 90TH0360-8, 1991, pp. 408-409.

[SHOO 1983] Shooman, M.L., "Software Engineering", Mc Graw - Hill Book Company, 1983.

[SIH 1993a] Sih, G.C., and Lee, E.A., "A compile-time scheduling Heuristic for Interconnection-Constrained Heterogeneous processor Architectures," IEEE Trans. Parallel and Distributed systems, vol. 4. No. 2, pp. 175-186, Feb. 1993.

[SIH 1993b] Sih, G. C., and Lee, E. A., "De-clustering: A New Multiprocessor Scheduling Technique", IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 6, June 1993, pp. 625-637.

[SINC 1987] Sinclair, J. B., "Efficient Computation of Optimal Assignment for Distributed Task", Journal of Parallel and Distributed Computing, Vol. 4, 1987, pp. 342-362.

[SING 1999] Singh, M.P., Kumar, V., and Kumar, A., "An Efficient Algorithm for Optimizing Reliability Index in Task-Allocation", Acta Ciencia Indica, Vol xxv m, 1999, pp. 437-444.

[SINH 2004] Sinha, Pradeep, K., "Distributed Operating Systems Concepts and Design", PHI, New Delhi, 2004.

[SITA 1995] Sitaram, B.R., "Distributed Computing – A User's View Point", CSI Communications, Vol. 18, No.10, 1995, pp. 26-28.

[SIVA 1988] Siva Ram Murthy, C., and Rajaraman, V., "Multiprocessor Architectures for Solving PDEs", Journals of the Institution of Electronics and Telecommunication Engineers, Vol. 34, No.3 1988, pp. 172-184.

[SIVA 1989] Siva Ram Murthy, C., and Rajaraman, V., "Task Assignment in a Multiprocessor System", Micro processing and Microprogramming, Vol. 26, 1989, pp. 63-71.

[SRIN 1999] Srinivasan, S., and Jha, N. K. "Safety and Reliability driven Task Allocation in Distributed Systems," IEEE Transactions on Parallel and Distributed Systems, Vol. 10, No. 3, Mar. 1999, pp. 238-251.

[STON 1977] Stone, H. S., "Multiprocessor Scheduling with the Aid of Network Flow Algorithms", IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, 1977, pp.85-93.

[STON 1978] Stone, H. S., "Critical Load Factors in Two Processor Distributed System", IEEE Transactions on Software Engineering, Vol. SE-4, No. 1, 1978, pp.254-258.

[STON 1987] Stone, H. S., (ed), "Introduction to Computer Architecture", Galgotia Publication Pvt. Ltd., New Delhi, 1987.

[TILL 1977] Tillman, F.A., Hwang, C.L., and Kua, W., "Determining Component Reliability and Redundancy for Optimum System

Reliability," IEEE Transactions on Reliability, Vol. R-26, 1977, pp. 162-165.

[TOWS 1986] Towsley, D., "Allocation Program Containing Branches and Loop within a Multiprocessor System", IEEE Trans. on Software Engineering, Vol. SE-12, No. 10, 1986, PP. 1018-1024.

[TURB 1972] Turban, E., "A sample survey of Operation Research Activities at the Corporate Level", Operation Research, Vol.20, 1072, PP. 708-721.

[VERM 1997] Verma, A. K., and Tamhankar, M. T., "Reliability-Based Optimal Task-Allocation in Distributed-Database Management Systems", IEEE Transactions on Reliability, Vol. 46, No. 4, 1997, pp. 452-459.

[WANG 2007] Wang, N, Lu, J. C., Kvam, P., "Reliability Modeling in Spatially Distributed Logistics Systems", IEEE Transactions on Reliability, Vol. 55, No. 3, 2006, pp. 525-534.

[WARD 1984] Ward, M. O., and Romero, D. J., "Assigning Parallel Executable Inter Communicating Subtasks to Processors", Proceeding of the 1984 International Conference on Parallel Processing, Bellaire, MI, 1984, pp. 392-394.

[WILL 1983] Williams, E., "Assigning Processes to Processors in Distributed Systems", Proceeding of the 1983 International Conference on Parallel Processing, 1983, pp. 404-406.

[WU 1980a] Wu, S. B., and Liu, M. T., "Assignment of Task and Resources for Distributed Processing", IEEE COMPCON fall 1980 Proceedings on Distributed Processing, 1980, pp. 665-672.

[WU 1980b] Wu, S. B., and Liu, M. T., "A Partition Algorithm for Parallel and Distributed Processing", Proc. of the 1980 International Conference on Parallel Processing, 1980, pp. 254-255.

[YADA 2002] Yadav, P.K., Singh, M.P. and Kumar, Harendra, " Task Allocation : An Algorithm for Systematic Allocation of tasks

in Distributed computing Environment" in :National Seminar on Current trends in Mathematics and computation held at Birla Institute of Technology, Ext. Center, NOIDA on December 1, 2005.

[YADA 2003] Yadav P. K., Sharma A. K. and Sharma Manisha "An Optimal Sequencing Strategy for Module Execution in Computer Communication System" published in Journal of Mathematical Science, Vol. 3, pp. 175- 203.

[YANG 1987] Yang, X. L., and Zhang, X. D., "A General Heuristic Algorithm of Task Allocation in Distributed Systems", Proc. of the 2nd International Conference on Computers and Applications, China, 1987, pp. 689-693.

[YEN 2007] Yen, Wei Chang, "A Simple Heuristic Algorithm for Generating all Minimal Paths," IEEE Transactions on Reliability, Vol. 56, No. 3, 2007, pp. 488-494.

[YUAN 2006] Yuan – Shun, Dai, Levitin, G., "Reliability and Performance of Tree Structured Grid Services", IEEE Transaction on Reliability, Vol. 55, No. 2, 2006, pp. 337-349.

[YUAN 2007] Yuan – Shun, Dai, Levitin, G., "Optimal Source Allocation for Maximizing Performance and Reliability in Tree Structured Grid Services", IEEE Transaction on Reliability, Vol. 56, No. 3, 2007, pp. 444-453.

[ZAHE 1991] Zahedi, F., and Ashrafi, N., "Software Reliability Allocation Based on Structure, Utility, Price, and Cost", IEEE Transactions on Software Engineering, Vol. 17, No. 4, April 1991, pp. 345-356.

[ZHOU 2007] Zhou, Zhibo, Zhou, Tong and Jinxiang Wang., "Performance of Multi-User DCSK Communication System Over Multipath Fading Channels, " IEEE Transactions on Communications, Vol. 55, No. 9, 2007, pp. 2478-2481.

GAMS JOURNAL OF MATHEMATICAL BIOSCIENCES

Vol. 3 No. 1 January-June 2007, pp. 88-96

Evaluation of Improved Reliability of Distributed Computing Systems Using Mathematical Programming Approach

Avanish Kumar,[1]* P.K. Yadav[2] and Mudit Bansal[1]

[1]Department of Mathematical Sciences and Computer Applications
Bundelkhand University, Jhansi - 284 128
[2]Research Planning and Business Development
Central Building Research Institute, Roorkee - 247 667

Abstract: Reliability is the extent to which a measurement procedure yields the same results on repeated trials. Without reliable measures, scientists cannot build or test theory and, therefore, cannot develop productive and efficient procedures for improving the quality of life. Reliability analysis for any distributed computing system is the current necessity and most important area of research. There are number of ways to improve the reliability of the distributed computing system. Optimising the reliability through task allocation is one of the problems in this category. Reliability is the assessment we make of how much measurement error we have experienced in processing our data. The problem presented in this article is based on the consideration of execution unreliability of the tasks to the processors. The main objective of this problem is to minimise the overall processing unreliability by allocating the tasks optimally to the processors of the distributed computing system using the method of matrix partitioning. Several sets of input data are considered to test the complexity and efficiency of the algorithm. It is found that the algorithm is suitable for arbitrary number of processors with the random program structures and workable in all the cases.

Keywords: Mathematical programming approach, Task allocation, Distributed computing systems.

Introduction

The main incentives for choosing distributed computing systems are higher throughput, improved reliability and better access to a widely communicated web of information. The increased commercialisation of communication computing systems means that ensuring system optimal

*Author for Correspondence. E-mail: dravanishkumar@yahoo.com

reliability is of critical importance. Inherently, a distributed computing system is more complex, therefore, it is very difficult to predict its performance. Mathematical modelling is the tool which plays an important role to predict the performance of distributed computing systems. Over the last several years, distributed computing systems have become popular as a very attractive option for fast computing and information processing. Distributed computing system is used to describe a system whenever there are several computers interconnected in some fashion so that a program or procedure runs on the system with multiple processors. However, it has different meanings to different systems because processors can be interconnected in many ways for various reasons. In the most general form, the word distribution implies that the processors are in geographically separate locations. Occasionally, it is also applied to an operation using multiple mini-computers, which are not a part of hardware, connected with each other and can also be connected through satellite. A user-oriented definition of distributed computing is stated [1, 2] as "The Multiple Computers utilised cooperatively to solve problems i.e. to process and maintain the large scale database of the programs which are to be executed on these types of systems". Some static [3-7] and dynamic [8-11] task allocation techniques have contributed a lot to development in software engineering as an independent area of research. Scheduling in general purpose distributed system [9] and dual processor and multi-processor scheduling with dynamic re-assignment [8] have drawn considerable attention of the researchers. Several methods have been reported in the literature, e.g. nteger programming [4, 12], critical delays consideration [13] and reliability evaluation to deal with various design and allocation issues in a distributed computing system. The study of complexity of the problem of assigning the modules to the processors in computing distributed computing ystems to minimise the total computational time has also discussed [9]. The other two similar lgorithms [14, 15] have been reported in the literature. Peng et al. [14] used matrix reduction echnique. According to the criteria given therein, a task is selected randomly to start with and hen assigned to a processor. Richard et al. [15] used branch and bound method for assignment nd scheduling communicating periodic tasks in a distributed computing system. Kim and Browne 16] gave an efficient method for optimal task allocations in distributed computing system owing o inter task communication effects. Due to numerous advances in the field of communication etworks, a wide variety of networks have come into existence. Ethernet is often used in local area etworks to allow independent devices to interconnect with one another within a relatively small hysical location [17, 18]. Asynchronous Transfer Mode has been proposed for wide area networks o integrate a variety of data communication services, including voice, video and plain text [18-0]. The high-performance parallel interface was originally developed to allow mainframes and upercomputers to communicate at a very high-speed [20]. A number of heuristic methods like near clustering [21, 22] and hierarchical clustering [23, 24] perform execution and scheduling to inimise the schedule length. Module redundancy has been used for fault detection and masking 25] by running two or more copies of the module. Module execution is done in a distributed omputing system dynamically. However, if each copy of the module is allowed to complete then ie degradation in throughput can be substantial. The main objective of this article is to maximise ie total program execution reliability by allocating the tasks in such a way that the allocated load n each processor should be balanced. For this purpose, we consider the execution unreliability f each task to each processor and design the allocation policy in such a way that total execution nreliability is minimised. The model utilised the mathematical programming technique for execution

of the module considering that each module is to be executed by the processor. The model addressed in this article is based on the consideration of execution unreliability (EUR) of the tasks to the processors. We suggested a modified method to assign all the tasks as per the required availability of processors so that none of the tasks remains unexecuted. Also, the present approach does not require adding dummy processors. The EURs presented by arrays in the form of execution unreliability matrix (EURM(,)) of order $m \times n$. The developed model is programmed in C and implemented. Several sets of input data are used to test the effectiveness and efficiency of the algorithm.

Objective

The objective of this problem is to maximise the total execution reliability by allocating the tasks optimally. For that purpose we considered the execution unreliability and used the method of matrix partitioning. This problem used a modified method to assign all the tasks as per the required availability of processors so that none of the tasks remains unexecuted even when the number of tasks is more than number of processors. The EUR is presented by arrays in the form of EURM(,) of the order $m \times n$. A procedure is then formulated to assign the tasks to the processors of the distributed computing systems based on the execution unreliability and is to be designed in such a way that the overall reliability is optimised under the pre-specified constraints.

Technique

For developing the technique we have formed the sub-problems using matrix partitioning i.e. each of the sub-problems is of the type of balanced assignment problem and devised a method to assign all tasks to the processors and formulating the unreliability function to measure EUR. Since the number of tasks is more than the number of processors, we divide the problem of unbalanced tasks assignment into sub-problems using matrix partitioning. This gives us balanced tasks assignment problems. First, we obtain the product of each row and each column except the position where the unreliability is zero (zero unreliability should be kept aside from the product of row or column) from the EURM(,) and store the results into Product_Row() and Product_Column(), each of them are one dimensional arrays. Select the first set of tasks (this set of tasks shall contain only as many tasks as the number of processors in the distributed computing system) on the basis of minimum unreliability against the tasks in the Product_Row() array. Store the result into EURM (, ,) a two-dimensional array. Repeat the process until remaining tasks are either less than or equal to the number of processors. If the tasks are equal to the number of processors of the computing system then it will become the last sub-problem, else to form the last problem we have to delete the column (processor) from EURM(, ,) on the basis of Product_Column( ) array i.e. this set shall contain only as many processors as number of tasks left, so that we delete the processors that have maximum unreliability in the Product_Column() array. For allocation purpose a modified version of row and column assignment method devised by Kumar et al. [20] is employed which allocates a task to a processor where it has minimum execution unreliability. For each sub-problem, calculate the execution unreliability of each processor and store the result in a linear array PEUR(j) where $j = 1, 2, \ldots, n$.

$$PEUR(j) = \prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} eur_{ij}\, x_{ij} \right\}$$

where $x_{ij} = \begin{cases} 1, \text{ if } i^{th} \text{ task is assigned to } j^{th} \text{ processor} \\ 0, \text{ otherwise} \end{cases}$

Algorithm

Consider a distributed computing system which consists of a set of n processor, $P = \{p_1, p_2, \ldots\ldots\ldots, p_n\}$ and a set of m executable tasks $T = \{t_1, t_2, \ldots\ldots\ldots, t_m\}$ which are to be processed by any one of the processors of the system.

Step 1

Input: m, n. EURM(,).

Step 2

Obtain the product of each row of the EURM(,) in such a way that, if any unreliability(ies) is (are) zero then keep it aside from product amount of that row (just to avoid the condition EUR* 0 = 0). Store the results in one-dimensional array Product_Row (,) of order m.

Step 3

Obtain the product of each column of the EURM(,), in such a way that if any unreliability(ies) is (are) zero then keep it aside from product of that column (just to avoid the condition EUR* 0 = 0). Store the results in one-dimensional array Product_Column (,) of order n.

Step 4

Partition the execution unreliability matrix EURM(,) of order $m \times n$ to sub-matrices such that the order of these matrices become square. Partitioning is to be made as follows:

Step 4.1

Select the n task on basis of Product_Row (,) array i.e. select the 'n' task corresponding to the most minimum product to next minimum product, if there is a tie select arbitrarily. (For the cases in which product is EUR * 0, minimum value depends only on EUR and the impact of zero is to be neglected.)

Step 4.2

Store the result in the two-dimensional array EURM(, ,) to form the sub-matrices of the sub-problems.

Step 4.3

If all the tasks are selected then go to step 4.7 else step 4.4.

Step 4.4

Repeat the steps 4.1 to 4.3 until the number of tasks become less than n.

Step 4.5
Select the remaining task say r, $r < n$, select the r processors on the basis of Product_Column (,) array i.e. the processors corresponding to the most maximum product to next maximum and if there is a tie, select arbitrarily (for the cases in which product is EUR * 0, maximum value depends only on EUR and the impact of zero is to be neglected).

Step 4.6
Store the result in the two-dimensional array EURM(, ,), which is the last sub-problem.

Step 4.7
List all the sub-problems formed through steps 4.1 to 4.6 and repeat steps 5 to 12 to solve each of these sub-problems.

Step 5
Find the minimum of each row of EURM(, ,) and replace it by 0.

Step 6
Find the minimum of each column of EURM(, ,) and replace it by 0.

Step 7
Search for a row in EURM(, ,), which has only one zero and assign the task(s) corresponding to this position. Add one to the counter i.e. nar = nar + 1 and store this position.

Step 8
Search for a column in EURM(, ,), which has only one zero and assign the task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position.

Step 9
Check whether nar = n, if not then pickup an arbitrary 0 and assign task(s) corresponding to this position. Add one to the counter i.e. nar = nar + 1 and also store this position, else, check column(s) position of 0's in unassigned row(s). Check the row(s) for any previous assignment in the corresponding column(s). Find the minimum of the entire elements for the remaining rows and replace it by zero.

Step 10
Evaluate Execution Unreliability.

Step 11
Execution reliability (Ereliability) is, thus, calculated as:
Ereliability = 1 – EUR

Step 12
Stop.

Implementation

Consider a system which consists of a set $T = \{t_1, t_2, t_3, t_4, t_5\}$ of 5 tasks and a set $P = \{p_1, p_2, p_3\}$ of 3 processors, where

Step 1

Input: 5, 3

$$\text{EMRU}(,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 0.03 & 0.04 & 0.06 \\ t_2 & 0.07 & 0.02 & 0.08 \\ t_3 & 0.02 & 0.09 & 0.06 \\ t_4 & 0.03 & 0.07 & 0.02 \\ t_5 & 0.08 & 0.05 & 0.04 \end{array}$$

Step 2

Obtain the product of each row and column of EURM(,), i.e. the products of each row and each column are as follows:

$$\text{Product\_Row}() = \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \\ 0.000072 & 0.000112 & 0.000108 & 0.000042 & 0.000160 \end{array}$$

$$\text{Product\_Column}() = \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.0000001008 & 0.0000002520 & 0.0000002304 \end{array}$$

Step 3

We partitioned the matrix EURM(, ,) to define the first sub-problem of EURM(, ,) by selecting rows corresponding to t_2, t_3, t_5 and second sub-problem of EURM(, ,) by selecting rows corresponding to the tasks t_1, t_4 and by deleting columns corresponding to p_1. The modified matrices for each sub-problem are:

Sub-problem I

$$\text{EURM}(1,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_2 & 0.07 & 0.02 & 0.08 \\ t_3 & 0.02 & 0.09 & 0.06 \\ t_5 & 0.08 & 0.05 & 0.04 \end{array}$$

Sub-problem II

$$\text{EURM}(2,) = \begin{array}{c|cc} & p_2 & p_3 \\ t_1 & 0.04 & 0.06 \\ t_4 & 0.07 & 0.02 \end{array}$$

Steps 4 and 5

We apply modified Hungarian method devised by Kumar et al. [20] to assign the tasks, on the basis of min $\{r_i\}$ and min $\{c_j\}$ from unreliability matrices for every i and j. We put $r_{ij} = 0$ and $c_{ij} = 0$, for every i and j. On applying this to all sub-problems, the modified matrices for each sub-problem are given as:

$$\text{EURM}(1,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_2 & 0.07 & 0.00 & 0.08 \\ t_3 & 0.00 & 0.09 & 0.06 \\ t_5 & 0.08 & 0.05 & 0.00 \end{array}$$

$$\text{EURM}(2,) = \begin{array}{c|cc} & p_2 & p_3 \\ t_1 & 0.00 & 0.06 \\ t_4 & 0.07 & 0.00 \end{array}$$

Steps 6, 7 and 8

After implementing assignment process, the allocation is thus obtained.

Tasks	→	Processors	EUR
t_1	→	p_2	0.04
t_2	→	p_2	0.02
t_3	→	p_1	0.02
t_4	→	p_3	0.02
t_5	→	p_3	0.04

Steps 9, 10 and 11

EUR = 0.0000000128
Ereliability = 0.9999999872

Step 12

Stop.

Conclusion

The algorithm presented in this article is capable of maximising the overall reliability of distributed computing systems through task allocation. In distributed computing system, tasks are allocated in such a way that their individual reliability of processing is optimised as well as it improves the overall reliability. In this approach, not only the loads of each processor get equally balanced but also none of the task gets unprocessed. This problem provides an optimal solution for assigning a set of m tasks of a program to a set of n processors where $m > n$ in a distributed computing system, i.e. to maximise the overall reliability of the system, the load of all allocated tasks on all the processors is equally balanced. The execution unreliability on different processors has been obtained.

Processors	p_1	p_2	p_3
Execution unreliability of each processor PEUR (,)	0.02	0.0008	0.0008
Total reliability [Ereliability(,)] i.e. (1-EUR(,))		0.9999999872	

The final result of the example is:

Tasks →	Processors	EUR	ER
t_3 →	p_1		
$t_1 \times t_2$ →	p_2	0.0000000128	0.9999999872
$t_4 \times t_5$ →	p_3		

This problem provides the optimal solution for improving the reliability and also deals with the performance on the basis of maximising reliability.

References

1. Wolman, A., Voelker, G. and Thekkath, C.A., 1994, "Latency Analysis of TCP on an ATM Network", *Proc. USENIX*, pp. 167-179.
2. Sitaram, B.R., 1965, "Distributed Computing: A User's View Point", *CSI Communications*, **18**, pp. 26-28.
3. Thekkath, C.A. and Levy, H.M., 1993, "Limits to Low-latency Communication on High-speed Networks", *ACM Trans. Computer Systems*, pp. 179-203.
4. Hung, C., Kasten, E.P. and Mckinley, P.K., 1994, "Design and Implementation of Multicast Operation for ATM-based High Performance Computing", *Proc. Supercomputing '94*, pp. 164-173.
5. Baca, D.F., 1989, "Allocation of Tasks to Processor in a Distributed System", *IEEE Trans. on Software Engineering*, **15**, pp. 1427-1436.
6. Tolmine, D. and Renwiek, J., 1993, "HiPPl Simplicity Yield Success", *IEEE Network*, pp. 28-32.
7. Sih, G.C. and Lee, E.A., 1993, "De Clustering: A New Multiprocessor Scheduling Technique", *IEEE Trans. on Parallel and Distributed Systems*, **4**, pp. 625-637.
8. Sih, G.C. and Lee, E.A., 1993, "A Compile-time Scheduling Heuristic for Interconnection-constrained Eterogeneous Processor Architectures", *IEEE Trans. on Parallel and Distributed Systems*, **4**, pp. 175-186.
9. Sagar, G. and Sarje, A.K., 1991, "Task Allocation Model for Distributed System", *Int. J. System Science*, **22(9)**, pp. 1671-1678.
10. Rotithor, H.G., 1994, "Taxonomy of Dynamic Task Scheduling in Distributed Computing Systems", *IEEE Proc. Computer Digit Tech.*, **14**, pp. 1-10.
11. Sinclair, J.B., 1987, "Efficient Computation of Optimal Assignments for Distributed Tasks", *J. Parallel Dist. Computer*, **4**, pp. 342-362.
12. Bhutani, K.K., 1994, "Distributed Computing", *The Indian Journal of Telecommunication*, pp. 41-44.
13. Dessoukiu-EI, O.I. and Huna, W.H., 1980, "Distributed Enumeration on Network Computers", *IEEE Trans. on Computer*, **29**, pp. 818-825.
14. Peng, D.-T., Abdel, K.G.S. and Zoher, T.F., 1997, "Assignment Scheduling Communicating Periodic Tasks in Distributed Real-time System", *IEEE Trans. on Software Engg.*, **13**, pp. 745-757.

15. Richard, R.Y., Lee, E.Y.S. and Tsuchiya, M., 1982, "A Task Allocation Model for Distributed Computer System", *IEEE Trans. on Computer*, **31**, pp. 41-47.
16. Kim, S.J. and Browne, J.C., 1998, "A General Approach to Mapping of Parallel Computations Upon Multiprocessor Architectures", *Proc. Int'l Conf. Parallel Processing*, University Park, Penn, **3**, pp. 1-8.
17. Bokhari, S.H., 1979, "Dual Processor Scheduling with Dynamic Re-assignment", *IEEE Trans. on Software Engineering*, **5**, pp. 341-349.
18. Nog, S. and Kotz, D., 1995, "A Performance Comparison of TCP/IP and MPI on FDDI", *Fast Ethernet, PCS-TR95-273*, Department of Computer Science, Dartmouth College.
19. Casavent, T.L. and Kuhl, J.G., 1988, "Taxonomy of Scheduling in General Purpose Distributed Computing System", *IEEE Trans. on Software Engineering*, **14**, pp. 141-154.
20. Kumar, V., Singh, M.P. and Yadav, P.K., 1995, "A Fast Algorithm for Allocation Task in Distributed Processing System", *Proceedings of the 30th Annual Convention of Computer Society of India*, Hyderabad, pp. 347-358.
21. Kumar, V., Singh, M.P. and Yadav, P.K., 1995, "An Efficient Algorithm for Allocating Tasks to Processors in a Distributed System", *Proceedings of the Nineteenth National System Conference (NSC-95)*, PSC College of Technology, Coimbatore, Organized by System Society of India, New Delhi, pp. 82-87.
22. Kumar, V., Singh, M.P. and Yadav, P.K., 1996, "An Efficient Algorithm for Multi-Processor Scheduling with Dynamic Re-assignment", *Proceedings of the Sixth National Seminar on Theoretical Computer Science*, Banasthali Vidyapith, pp. 105-118.
23. Kumar, V., Yadav, P.K. and Bhatia, K., 1998, "Optimal Task Allocation in Distributed Computing Systems Owing to Inter Task Communication Effects", *Proc. of CSI-98: IT for the New Generation*, pp. 369-378.
24. Kohler, W.H., 1975, "A Preliminary Evaluation of the Critical Path Method for Scheduling Tasks on Multiple Processor Systems", *IEEE Trans. on Computer*, **24**, pp. 1235-1238.
25. Chu, W.W., 1969, "Optimal File Allocation in a Multiple Comp. Systems", *IEEE Trans. on Comp.*, **18**, pp. 885-889.

Evaluation of Optimal Executiontime of Computer System by exhaustive search approach

The Allocation problems in any computer system play the key role for deciding the performance of the system. The allocation put the direct impact of software resources as well as hardware resources. In heterogeneous Computer systems, partitioning of the application software in to module and the proper allocation of these modules dissimilar processors are important factors, which determine the efficient utilization of resources. The static model discussed in this paper provide an optimal solution for assigning a set of "m" modules of a task to a set of "n" processors where m > n in a computer system by using an Exhaustive Search Approach for evaluation for optimal time of the system. The Impact of Communication Time has also been taken into consideration.

DR. P.K. YADAV*, DR.AVANISH KUMAR** & MUDIT BANSAL**

Abstract:

The Allocation problems in any computer system play the key role for deciding the performance of the system. The allocation put the direct impact of software resources as well as hardware resources. In heterogeneous Computer systems, partitioning of the application software in to module and the proper allocation of these modules dissimilar processors are important factors, which determine the efficient utilization of resources. The static model discussed in this paper provide an optimal solution for assigning a set of "m" modules of a task to a set of "n" processors where m > n in a computer system by using an Exhaustive Search Approach for evaluation for optimal time of the system. The Impact of Communication Time has also been taken into consideration. Approach suggested in the paper defines an index based on processing time along with the communication time of the computer system. The Execution time of the tasks has been taken in the form for matrix of order m x n named as ETM (,) and communication time has been also been taken in the form of symmetric matrix of order m named as CTM (,). The several sets of input data are considered to test the complexity and efficiency of the algorithm. It is found that the algorithm is suitable for arbitrary number of processor with the random program structure and workable in all the cases.

Keywords:

Optimization, Computer System, Execution Time, Task Allocation.

(1) Introduction:

Task allocation is the process of partitioning a set of programming modules into a number of processing groups, known as tasks, where each group executes on a separate processor. The general allocation problem is NP-complete. To obtain an easy solution, it is essential to make use of heuristics approach that provides a near optimal solution in a reasonable amount of time. Hungarian approach is suggested for solving the assignment problem, but this approach is to be applicable only to m-tasks to the m-processors i.e. for balanced Assignment problem. For the unbalanced assignment problem where m > n, the Hungarian method suggest to add the dummy tasks/processors to make the effectiveness matrix square. Distributed computing systems have homogenous and/or heterogeneous processors that are connected through a communication network. It provides the capability for the utilization of remote computing resources and allow for increased level of flexibility, reliability, and modularity. Assignment in distributed system has some major advantage. Computer systems are being converted from host-centralized system to distributed systems, represented by client server model, through corporate efforts in downsizing and rightsizing. People expect much more extendibility of distributed system as workstation and personal computers are becoming more powerful and inexpensive, application which run on distributed system are increasing, and the number of users also increasing. The best-known research problem for such systems is the allocation problem, in which either system reliability is maximized or total system cost is minimized. These problems may be categorized in static [1,5,10-12,17,20,21-22,24-25] and dynamic [3-4,9,13,] assignment problems. Several other methods have been reported in the literature, such as, Integer programming [7], Branch and bound technique [18], Matrix reduction technique [20], Reliability optimization [15-16], Modeling [2,8]. The series parallel redundancy-allocation problem has been studied with different approaches, such as, Non-linear techniques [23], and Heuristic techniques [6].

*RPBD, Central Building Research Institute, Roorkee (UA)

**Department of Matehmatical Sciences & Computer Applications, Bundelkhand University, Jhansi (UP)

The present work suggests an exhaustive search approach for the allocation problem through index optimization. The index is based on the time of the tasks to the processors for their execution to processors and also time of communication amongst the tasks.

(2) Definitions :

Execution Time :

Each task t_i has an Execution Time [ET] ET_{ij} ($1 <= i <= m$ and $1 <= j <= n$), which is the time of processing the task t_i on the processor p_j.

Communication Time :

The Communication Time [CT] CT_{ij} ($1 <= i <= m$ and $1 <= j <= m$) of the interacting tasks t_i and t_j is incurred due to the data units exchanged between them during the process of execution.

Distributed Systems :

A distributed system is characterized by having processors functions, the database, or system control physically dispersed and interconnected by communication facilities.

Assignments :

Assignment is a process in which any types of activity perform to any type of resource and time for a resource to complete an activity could be considered the effectiveness associated with using a given type of resource on a given activity.

(3) Assumptions :

The completion of a program from computational point of view means that all related tasks have got executed and the final output has been generated after integrating the respective outputs of individual programs.

Some of the tasks have restrictions for being allocated on some particular processor(s) because of the non-availability of desired facilities, which may include, access to particular peripheral device, and high-speed arithmetic capability of the processor. The execution time of such tasks is taken to be infinite, on those processors where these tasks cannot be executed.

Whenever two or more tasks are assigned to the same processor, the communication time between them is assumed to be zero.

The number of tasks to be allotted is more than the number of processors.

(4) Objective :

Let the given system consists of a set $P = \{p_1, p_2, p_3, p_4, ..., p_n\}$ of n processors, interconnected by communication links and a set $T = \{t_1, t_2, t_3, t_4, ..., t_m\}$ of m tasks. The processing execution time of individual tasks corresponding to each processor are given in the form of matrix ETM (,) order m x n. The communication time is taken in the square symmetric matrices CTM (,) of order n respectively. The functions to measure ET, and CT are then formulated. A procedure to assign the tasks to the processors of the distributed systems based on execution time is to be designed in such a way that the overall time is to be optimizing under the pre specified constraints. The load on each processor has been also taken care off in order to balance the load of processor.

(5) Technique :

Let the given system consists of a set $P = \{p_1, p_2, p_3, p_4, ..., p_n\}$ of n processors, interconnected by communication links and a set $T = \{t_1, t_2, t_3, t_4, ..., t_m\}$ of m tasks. The processing execution time of individual tasks corresponding to each processor are given in the matrix ETM (,) of order m x n. The communication time is taken in the square symmetric matrix CTM (,) of order n. Initially, we obtain the task combinations in order to make the set of task(s) equals to number of processor. These combinations shall be $n * {}^{m}C_{m-n}$ (= nl, say) and to be store in TCOMB (). Then we obtained an index, which is based on the time of the tasks to the processors for the execution of tasks to various processors, and also time of communication amongst the task to each of the combination, the maximum value of the index shall give the optimal result. The assignment of tasks to processors may be done in different ways. To allocate the task to one of the processors, the minimum value of each row and column of ETM (,) is obtained. Let min $\{r_i\}$ represent the minimum row cost value corresponding to the tasks t_i and min $\{c_j\}$ represent the minimum column time value for processor p_j. These values are then replaced to 0 in ETM (,). For allocation purpose a modified version of row and column assignment method of Kumar et al [14] is employed which allocates a task to a processor where it has minimum execution time. The overall assignment time [Etime] is expressed as the sum of execution time and communication time of all the tasks as follows:

$$Etime = \left[\sum_{i=1}^{n}\left\{\sum_{j=1}^{n} ET_{ij}x_{ij}\right\} + \sum_{j=1}^{n}\left\{\sum_{j=1}^{n} CT_{ij}y_{ij}\right\}\right]$$

where, $x_{ij} = \begin{cases} 1, \text{if } i^{th} \textit{ task is assigned to } j^{th} \text{ processor} \\ 0, \text{otherwise} \end{cases}$, and

$y_{ij} = \begin{cases} 1, \text{if the } \textit{task assigned} \text{ to processor i communicate with the task assigned to processor j} \\ 0, \text{otherwise} \end{cases}$

Index = (Etime)$^{-1}$

(6) Implementation :

Example-I

Consider a system consisting of a set $T = \{t_1, t_2, t_3, t_4\}$ of 4 tasks and a set $P = \{p_1, p_2, p_3\}$ of 3 processors,

Step-1: Input: 4, 3

$$ETM\ (,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 8 & 12 & 7 \\ t_2 & 9 & 8 & 11 \\ t_3 & 12 & 9 & 6 \\ t_4 & 10 & 11 & 12 \end{array}$$

$$CTM(,) = \begin{array}{c|cccc} & t_1 & t_2 & t_3 & t_4 \\ t_1 & 0 & 3 & 6 & 9 \\ t_2 & 3 & 0 & 4 & 5 \\ t_3 & 6 & 4 & 0 & 7 \\ t_4 & 9 & 5 & 7 & 0 \end{array}$$

Step-2:

$$TCOMB(,) = \begin{bmatrix} 3 & 4 & 2 & 4 & 2 & 3 & 1 & 4 & 1 & 3 & 1 & 2 \\ 4 & 3 & 4 & 2 & 3 & 2 & 4 & 1 & 3 & 1 & 2 & 1 \end{bmatrix}$$

Step-3:

$$TCOMB(1) = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Step-3.1-3.2:

$$ETM(,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_3 * t_4 & 22 & 20 & 18 \end{array}$$

$$NCTM(,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 0 & 3 & 6 \\ t_2 & 3 & 0 & 4 \\ t_3 * t_4 & 6 & 4 & 0 \end{array}$$

$$NETM(,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 8 & 12 & 7 \\ t_2 & 9 & 8 & 11 \\ t_3 * t_4 & 22 & 20 & 18 \end{array}$$

Step-4 & 5:

On applying modified Hungarian method devised by Kumar et al [14] to assign the task, min $\{r_i\}$ from NETM (,) for every i, $r_{13} = 7$, $r_{22} = 8$, $r_{33} = 18$. Making $r_{13} = r_{22} = r_{33} = 0$. Again min $\{c_j\}$ from NETM (,) for every j, are $c_{11} = 8$, $c_{22} = 0$, $c_{33} = 0$. Making $c_{11} = 0$, so that, we get,

$$NETM(,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 0 & 12 & 0 \\ t_2 & 9 & 0 & 11 \\ t_3 * t_4 & 22 & 20 & 0 \end{array}$$

Step-6, 7 & 8:

After implementing assignment process, the first set of the allocation is thus obtained.

Tasks	-	Processors	ET	CT
t_1	-	p_1	08	03
t_2	-	p_2	08	06
$t_3 * t_4$	-	p_3	18	04

Step-9:

ETT (1) := 34

Step-10:

CTT (1) := 13

Step-11:

Etime (1) := 47

Step-12:

Index (1) := 0.02127650

Step-13:

On repeating the above process, the assignments and their corresponding related values of ET, CT, Etime, and Indexes are thus obtained, which are shown in the following table-1:

Table-1: Assignment Table

S. No.	ET	CT	E time	INDEX	ASSIG-I	ASSIG-II	ASSIG-III
1	34	13	47	0.02127650	t_1 p_1	t_2 p_2	$t_3 * t_4$ p_3
2	34	17	51	0.01960780	t_1 p_1	t_2 p_2	$t_4 * t_3$ p_3
3	33	13	46	0.02173910	t_1 p_1	t_3 p_3	$t_2 * t_4$ p_2
4	33	22	55	0.01818181	t_1 p_1	t_3 p_3	$t_4 * t_2$ p_2
5	34	17	51	0.01960780	t_1 p_3	t_4 p_1	$t_2 * t_3$ p_2
6	34	22	56	0.01785710	t_1 p_3	t_4 p_1	$t_3 * t_2$ p_2
7	32	13	45	0.02222222	t_2 p_2	t_3 p_3	$t_1 * t_4$ p_1
8	32	16	48	0.02083333	t_2 p_2	t_1 p_3	$t_4 * t_1$ p_1
9	31	17	48	0.02083333	t_1 p_2	t_4 p_1	$t_1 * t_3$ p_3
10	31	16	47	0.02127650	t_2 p_2	t_4 p_1	$t_3 * t_1$ p_3
11	34	16	50	0.02000000	t_1 p_3	t_4 p_2	$t_2 * t_1$ p_1
12	34	22	56	0.01785710	t_1 p_3	t_4 p_2	$t_2 * t_3$ p_1

Step-14:

Stop.

Example-II

The results of a system which consist a set T = $\{t_1, t_2, t_3, t_4, t_5\}$ of 5 tasks and a set P = $\{p_1, p_2, p_3\}$, of 3 processors where,

$$ETM(,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 8 & 12 & 7 \\ t_2 & 9 & 8 & 11 \\ t_3 & 12 & 9 & 6 \\ t_4 & 10 & 11 & 12 \\ t_5 & 7 & 6 & 2 \end{array} \qquad CTM(,) = \begin{array}{c|ccccc} & t_1 & t_2 & t_3 & t_4 & t_5 \\ t_1 & 0 & 3 & 6 & 9 & 8 \\ t_2 & 3 & 0 & 4 & 5 & 6 \\ t_3 & 6 & 4 & 0 & 7 & 9 \\ t_4 & 9 & 5 & 7 & 0 & 5 \\ t_5 & 8 & 6 & 9 & 5 & 0 \end{array}$$

The following tables shows the results as obtain after implementing the present algorithm

Tasks	→	Processors	ET	CT
$t_1 * t_4$	→	p_1	18	3
t_2	→	p_2	8	6
$t_3 * t_5$	→	p_3	8	4

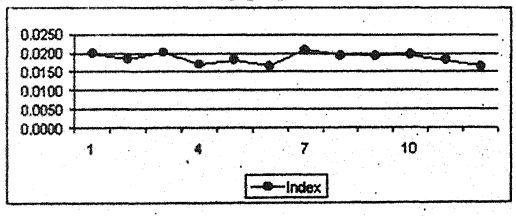
Tasks	→	Processors	Etime	Index
$t_1 * t_4$	→	p_1		
t_2	→	p_2	47	0.021276595
$t_3 * t_5$	→	p_3		

(7) Conclusions :

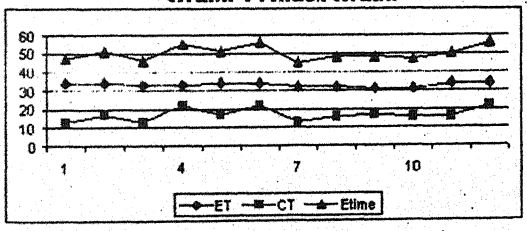
The present paper discusses an assignment problem through optimization techniques. In this paper we have chosen the problem in which number of tasks are more than the number of processors of the distributed system. The present method deals the case when the index is based on the time of the tasks to the processors for the execution of tasks to various processors and also communication time amongst the tasks. The method is presented in computational algorithmic form and implemented on the several sets of input data to test the performance and effectiveness of the algorithm. The optimal result of the Example-I giving in the paper are shown in the following table:

Tasks	→	Processors	Etime	Index
t_3	→	p_3		
t_2	→	p_2	45	0.02222222
$t_1 * t_4$	→	p_1		

The graphical representation of the results mentioned in Table-1 of Step-12 of the implementation part of this paper are shown in the following graphs:



Graph-1 : Index Graph



Graph-2 : Execution Time Graph

The following table shows the results of Example-II given in its implementation part of 8.2 in this paper, as obtain after implementing the present algorithm:

Tasks	→	Processors	Ettme	Index
$t_1 * t_4$	→	p_1		
t_2	→	p_2	47	0.0212765
$t_3 * t_5$	→	p_3		

References:

(1) Baca, D.F., "Allocation Modules to Processor in a Distributed System", IEEE Transactions on Software Engineering, vol. 15, pp. 1427-1436, 1989. (2) Bierbaum, Rene L., Brown, Thomas D. and Kerschen, Thomas J., "Model-Based Reliability Analysis", IEEE Transactions on Reliability, vol. 51, pp. 133-140, 2002. (3) Bokhari, S.H., "Dual Processor Scheduling with Dynamic Re-Assignment", IEEE Transactions on Software Engineering, vol. 5, pp. 341-349,1979. (4) Casavent, T.L. and Kuhl, J. G., "A Taxonomy of Scheduling in General Purpose Distributed Computing System", IEEE Transactions on Software Engineering, vol. 14, pp. 141-154, 1988. (5) Chu, W.W., "Optimal File Allocation in a Multiple Computing System", IEEE Transactions on Computer, vol. 18, pp. 885-889,1969. (6) Coit, D.W. and Smith, A.E., "Reliability Optimization of Series Parallel Systems using a Genetic Algorithm", IEEE Transactions on Reliability, vol. 45, pp. 254-260,1996. (7)Dessoukiu-El, O.I. and Huna, W. H., "Distributed Enumeration on Network Computers", IEEE Transactions on Computer, vol. C-29. pp. 818-825, 1980. (8) Fitzgerald, Kent, Latifi, Shahram and Srimani, Pradip K., "Reliability Modeling and Assessment of the Star-Graph Networks", IEEE Transactions on Reliability, vol. R-51, pp. 49-57, 2002. (9) Kumar, Avanish, "Optimizing for the Dynamic Task Allocation", published to the proceedings of the III Conference of the International Academy of Physical Sciences held at Allahabad, pp. 281-294, 99. (10) Kumar, Avanish, "An Algorithm for Optimal Index to Tasks Allocation Based on Reliability and cost", published to the proceedings of International Conference on Mathematical Modeling held at Roorkee, pp. 150-155, 2001. (11) Kumar, V. Singh, M. P. and Yadav, P.K., "An Efficient Algorithm for Allocating Tasks to Processors in a Distributed System", Proc. of the 19th National system conference, SSI, held at Coimbatore, India pp. 82-87, 1995. (12) Kumar, V. Singh, M.P. and Yadav, P.K., "A Fast Algorithm for Allocating Tasks in Distributed Processing System", Proc. of the 30th Annual Convention of CSI, held at Hyderabad, India pp. 347-358,1995. (13) Kumar, V. Singh, M.P. and Yadav, P.K., "An Efficient Algorithm for Multi-processor Scheduling with Dynamic Reassignment", Proc. of the 6th National seminar on theoretical Computer Science, held at Banasthally Vidyapeeth, India pp. 105-118, 1996. (14) Kumar, V. Yadav, P.K. and Bhatia, K., "Optimal Task Allocation in Distributed Systems owing to Inter Tasks Communication Effects", Proc. of the 33rd Annual convention of system society of India, held at New Delhi, India pp. 369-378,1998. (15) Lin, Min-Sheng, "A Linear-time Algorithm for Computing K-terminal Reliability on Proper Interval Graphs", IEEE Transactions Reliability, vol. 51, pp. 58-62, 2002. (16) Lyu, Michael R., Rangarajan, Sampath and Moorsel, Aad P. A. Van, "Optimal Allocation of test Resources for Software Reliability growth modeling in Software Development", IEEE Transactions on Reliability, vol. R-51, pp. 183-192, 2002. (17) Peng, Dar-Tezen, Shin, K. G. and Abdel, Zoher T. F., "Assignment Scheduling Communication Periodic Tasks in Distributed real time System", IEEE Transactions on software Engg. vol. SE-13, pp. 745-757,1997. (18) Richard R.Y., Lee, E.Y.S. and Tsuchiya, M., "A Task Allocation Model for Distributed Computer System", IEEE Transactions on Computer, vol. C-31, pp. 41-47,1982. (19) Rotithor, H.G., "Taxonomy of Dynamic Task Scheduling in Distributed Computing Systems", IEEE Proc. Computer Digit Tech., vol. 14, pp. 1-10,1994. (20) Sagar, G., and Sarje, A.K., "Task Allocation Model for Distributed System", Int. J. System Science, vol. 22, pp. 1671-1678,1991. (21) Singh, M.P., Kumar, V. and Kumar, A., "An Efficient Algorithm for Optimizing Reliability Index in Tasks-Allocation", Acta Ciencia Indica, vol. xxv m, pp. 437-444,1999. (22) Srinivasan, Santhanam and Jha. K. Niraj, "Safety and Reliability Driven Task Allocation in Distributed System", IEEE Transactions on Parallel and Distributed Systems, vol. 10, pp. 238-250,1999. (23) Tillman, F. A., Hwang, C. L. and Kuo, W., "Determining Component Reliability and Redundancy for Optimum System Reliability", IEEE Transactions on Reliability, vol. R-26, pp. 162-165,1977. (24) Yadav, P. K. and Kumar, Avanish, "An Efficient Static Approach for Allocation through Reliability Optimization in Distributed Systems", presented at the International conference on Operations Research for Development (ICORD2002) held at Chennai, 2002., (25) Zahedi, E., and Ashrafi, N., "Software Reliability Allocation based on Structure, Utility, Price and Cost", IEEE Transactions on Software Engineering, vol. -17, pp. 345-356,1991.